



```

CCCCCCCC 000000 BBBB BBBB AAAA AAAA CCCCCC CCCCCC EEEEEEEEE PPPPPPP TTTTTTTTT
CCCCCCCC 000000 BBBB BBBB AAAA AAAA CCCCCC CCCCCC EEEEEEEEE PPPPPPP TTTTTTTTT
CC         00         00 BB         BB AA         AA CC         CC         EE         PP         PP         TT
CC         00         00 BB         BB AA         AA CC         CC         EE         PP         PP         TT
CC         00         00 BB         BB AA         AA CC         CC         EE         PP         PP         TT
CC         00         00 BB         BB AA         AA CC         CC         EE         PP         PP         TT
CC         00         00 BBBB BBBB AA         AA CC         CC         EEEEEEE PPPPPPP TT
CC         00         00 BBBB BBBB AA         AA CC         CC         EEEEEEE PPPPPPP TT
CC         00         00 BB         BB AAAAAAAAAA CC         CC         EE         PP         TT
CC         00         00 BB         BB AAAAAAAAAA CC         CC         EE         PP         TT
CC         00         00 BB         BB AA         AA CC         CC         EE         PP         TT
CC         00         00 BB         BB AA         AA CC         CC         EE         PP         TT
CCCCCCCC 000000 BBBB BBBB AA         AA CCCCCC CCCCCC EEEEEEEEE PPPPPPP TT
CCCCCCCC 000000 BBBB BBBB AA         AA CCCCCC CCCCCC EEEEEEEEE PPPPPPP TT

LLLLLLLLLLLLLLLLLLLL I I I I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I I I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I I I I SSSSSSSS
LLLLLLLLLLLLLLLLLLLL I I I I I SSSSSSSS

```



```
0001 0 %TITLE 'COBSACCEPT - VAX COBOL ACCEPT Statement'
0002 0 MODULE COBSACCEPT (
0003 0 IDENT = '1-018' ) = ! File: COBACCEPT.B32 EDIT:LGB1018
0004 0
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1
0031 1 ++
0032 1 FACILITY: COBOL SUPPORT
0033 1
0034 1 ABSTRACT:
0035 1
0036 1 Supports the COBOL ACCEPT statement.
0037 1
0038 1 Contains COB$$OPEN_IN to open an RMS file for input.
0039 1
0040 1
0041 1 ENVIRONMENT: VAX-11 User Mode
0042 1
0043 1 AUTHOR: Rich Reichert, CREATION DATE: 16-JULY-79
0044 1
0045 1 MODIFIED BY:
0046 1
0047 1 1-001 - Original. RKR 16-JULY-79
0048 1 1-002 - Make COB$$OPEN_IN stop instead of signal on open error.
0049 1 RKR 4-SEPT-79
0050 1 1-003 - Make COB$$READ_RMS signal COB$_EOFON_ACC if an EOF is
0051 1 encountered during reading.
0052 1 Do string copy into caller's buffer via CH$COPY instead of
0053 1 STR$COPY to avoid dependency on STR$ routines.
0054 1 RKR 14-SEPT-79
0055 1 1-004 - Identify file name on bad RMS status other than EOF.
0056 1 RKR 25-SEPT-79
0057 1 1-005 - Change name of symbolic LIBRARY file. RKR 1-OCT-79
```



```
.. 58      0058 1 1-006 - Make module name match entry point. RKR 20-OCT-79
.. 59      0059 1 1-007 - Change references to LIB$ INVARG to COB$ INVARG.
.. 60      0060 1 1-008 - Make sensitive to names in REQUIRE file. RKR 21-OCT-79
.. 61      0061 1 1-009 - Improve errors signaled. RKR 21-OCT-79
.. 62      0062 1      Cosmetic changes. RKR 21-OCT-79
.. 63      0063 1 1-010 - Imperative clean-ups, also try SYS$ logicals.
.. 64      0064 1      PDG 00-FEB-81
.. 65      0065 1 1-011 - Fix call to $STRNLOG to test for SS$ NORMAL
.. 66      0066 1      (since SS$ NOTRAN is a success status).
.. 67      0067 1      Add COBSACCEPT EOF to allow ACCEPT ... AT END imp-statement.
.. 68      0068 1      Allow MAX(COBSR_ACC_SIZE, .STRING[DS$W_LENGTH]) bytes for ACCEPT.
.. 69      0069 1      PDG 24-Jul-1981
.. 70      0070 1 1-012 - Updated copyright date. LB 9-Aug-81
.. 71      0071 1 1-013 - Removed COBSACCEPT EOF. This functionality is provided by a flag
.. 72      0072 1      passed to COBSACCEPT.
.. 73      0073 1 1-014 - Add code in COBSACCEPT to check the STV2 field in the RAB to
.. 74      0074 1      determine if the terminator is an escape sequence, and if so,
.. 75      0075 1      to return the escape sequence in the user's buffer. This was
.. 76      0076 1      done in response to an SPR regarding incompatibilities between
.. 77      0077 1      COBOL-74 and VAX-11 COBOL. LEB 16-FEB-82
.. 78      0078 1 1-015 - Version 3 ACCEPT with screen enhancements. LGB 15-AUG-83
.. 79      0079 1 1-016 - Code converted from QIO calls to RMS. LGB 20-JAN-84
.. 80      0080 1 1-017 - Reset RAB[RAB$V ETO] bit in COBSACCEPT.
.. 81      0081 1      Added code to COB$$ILLEGAL_TERM and COB$$DELETE_KEY to handle
.. 82      0082 1      a Control Z situation.
.. 83      0083 1      Added a condition to the IF statement in COB$$ILLEGAL_TERM that
.. 84      0084 1      handles bell ringing for an illegal terminator.
.. 85      0085 1      Put an RMS workaround in routine COB$$PARTIAL_SEQ. LGB 11-JUL-84
.. 86      0086 1 1-018 - Bug fix to COBSACCEPT - reverse IF stmt within the CH$COPY stmt.
.. 87      0087 1      LBG 10-SEP-84
.. 88      0088 1 1--
```



```

90 0089 1  !
91 0090 1  ! PROLOGUE FILE
92 0091 1  !
93 0092 1  ! REQUIRE 'RTLIN:COBPROLOG' ;
94 1609 1  ! Switches, Psects, Include
95 1610 1  ! files
96 1611 1  ! TABLE OF CONTENTS:
97 1612 1  !
98 1613 1  ! FORWARD ROUTINE
99 1614 1  ! COBSACCEPT,
100 1615 1  ! COBSACC_SCR,
101 1616 1  ! COBSACC_SCR_FILE,
102 1617 1  ! Perform ACCEPT
103 1618 1  ! COBS$OPEN IN : NOVALUE,
104 1619 1  ! COBS$RMS_GET : NOVALUE,
105 1620 1  ! COBS$RMS_PUT_BYTE : NOVALUE,
106 1621 1  ! COBS$RMS_PUT_BUFFER : NOVALUE,
107 1622 1  ! COBS$CONTROL_Z : NOVALUE,
108 1623 1  ! COBS$PARTIAL_SEQ : NOVALUE,
109 1624 1  ! COBS$DELETE_KEY : NOVALUE,
110 1625 1  ! COBS$ILLEGAL_TERM : NOVALUE,
111 1626 1  ! COBS$CLEAN_UP : NOVALUE,
112 1627 1  ! COBS$RPG_CLEAN_UP : NOVALUE,
113 1628 1  ! COBS$FORMAT_FOUR ;
114 1629 1  ! Perform ACCEPT
115 1630 1  ! EQUATED SYMBOLS:
116 1631 1  !
117 1632 1  ! LITERAL
118 1633 1  ! NUM_UNITS = COBSK_UNIT_MAX - COBSK_UNIT_MIN + 1 ;
119 1634 1  ! Number of units
120 1635 1  ! LITERAL
121 1636 1  ! DISP = 0,
122 1637 1  ! DNA = 1,
123 1638 1  ! POS = 2,
124 1639 1  ! POS_DNA = 3,
125 1640 1  ! ACC_ADV = 4,
126 1641 1  ! ACC_DNA = 5,
127 1642 1  ! FLAG_MASK = 15,
128 1643 1  ! Code for Display
129 1644 1  ! V_BELL = 16,
130 1645 1  ! V_CONV = 32,
131 1646 1  ! V_DEC_PT = 64,
132 1647 1  ! Code for Display no Advancing
133 1648 1  ! V_NO_SIGN = 128,
134 1649 1  ! V_PROTECT = 256,
135 1650 1  ! V_NO_ECHO = 512,
136 1651 1  ! V_ADV = 1024,
137 1652 1  ! Code for Positioning
138 1653 1  ! V_COB_RPG = 2048,
139 1654 1  ! Code for Positioning no Advancing
140 1655 1  ! DEL_KEY = %X'7F',
141 1656 1  ! CZ = %X'1A',
142 1657 1  ! CARR_RET = 0,
143 1658 1  ! LINE_FD = 1,
144 1659 1  ! RING_BELL = 2,
145 1660 1  ! Parameters for routine
146 1661 1  ! RMS_HEADER = 14 ;
147 1661 1  ! COBS$RMS_PUT_BYTE
148 1661 1  !
149 1661 1  ! RMS uses up 14 bytes of
150 1661 1  ! the buffer for header info
```



```
147 1662 1 : GUARDS:
148 1663 1 :
149 1664 1 : Since the code assumes that COBSK_UNIT_MIN equals 0, and COB_TABLE
150 1665 1 : has only 7 items in it, we safeguard this module.
151 1666 1 :
152 1667 1 %IF COBSK_UNIT_MIN NEQ 0 %THEN %ERROR('Unexpected COBSK_UNIT_MIN value') %FI
153 1668 1 %IF COBSK_UNIT_MAX GTR 6 %THEN %ERROR('Unexpected COBSK_UNIT_MAX value') %FI
154 1669 1 :
155 1670 1 GLOBAL
156 1671 1 ACC_SCR : INITIAL (0) ; ! Flag for COBSACCEPT
157 1672 1 OWN
158 1673 1 XABTRM : $XABTRM_DECL, ! RMS XABTRM Control Block
159 1674 1 XAB_ITMLST : $ITMLST_DECL (ITEMS=2), ! Item list for XABTRM
160 1675 1 :
161 1676 1 :+
162 1677 1 : Terminator mask.
163 1678 1 : Leave Control C, Y, S, and Q for VMS.
164 1679 1 : Control I is Tab. Control M is Carriage Return.
165 1680 1 : ENTER key has the same value as <CR> i.e. Control M.
166 1681 1 : Escape key is a terminator for a VMS V4 system.
167 1682 1 : Delete key is processed via routine COB$$$DELETE_KEY.
168 1683 1 :-
169 1684 1 :
170 1685 1 MASK_VECTOR : BITVECTOR [160] ! 20 bytes by 8 bits
171 1686 1 PRESET ( [TAB] = 1,
172 1687 1 [CR] = 1,
173 1688 1 [26] = 1, ! Control z
174 1689 1 [01] = 1, ! Control a
175 1690 1 [02] = 1, ! Control b
176 1691 1 [04] = 1, ! Control d
177 1692 1 [05] = 1, ! Control e
178 1693 1 [06] = 1, ! Control f
179 1694 1 [07] = 1, ! Control g
180 1695 1 [08] = 1, ! Control h
181 1696 1 [10] = 1, ! Control j
182 1697 1 [11] = 1, ! Control k
183 1698 1 [12] = 1, ! Control l
184 1699 1 [14] = 1, ! Control n
185 1700 1 [15] = 1, ! Control o
186 1701 1 [16] = 1, ! Control p
187 1702 1 [18] = 1, ! Control r
188 1703 1 [20] = 1, ! Control t
189 1704 1 [21] = 1, ! Control u
190 1705 1 [22] = 1, ! Control v
191 1706 1 [23] = 1, ! Control w
192 1707 1 [24] = 1, ! Control x
193 1708 1 [27] = 1, ! Escape key for Arrow & PF Keys
194 1709 1 ! and Alternate Keypad Mode
195 1710 1 ! Delete key
196 1711 1 [127] = 1, ! SS3 for Professional Keys
197 1712 1 [143] = 1, ! CSI for Professional Keys
198 1713 1 [155] = 1,
199 1714 1 ) ;
200 1715 1 :
201 1716 1 : MACROS:
202 1717 1 :
203 1718 1 MACRO COB$$$STVO_TERM = 12,0,8,0 %; ! Location of terminator if it
```



```

204      1719      1      MACRO COB$$B_STV2_LEN = 14,0,8,0 %;      ! is not an escape sequence
205      1720      1      ! Length of escape sequence
206      1721      1
207      1722      1      MACRO
208      1723      1      $VERIFY_TERMINATOR =
209      1724      1      +
210      1725      1      If parameter KEY not sent (.KEY = 0) then CR, TAB, CONTROL Z,
211      1726      1      and DELETE KEY are the only legal terminators.
212      1727      1      -
213      1728      1      If parameter KEY not 0 then CR, TAB, CONTROL Z, DELETE KEY, PF,
214      1729      1      ARROW and SPECIAL FUNCTION PROFESSIONAL Keys are legal terminators.
215      1730      1      Copy terminator to KEY parameter if valid.
216      1731      1      Flag LEGAL set to 1 if terminator is valid.
217      1732      1      -
218      1733      1      BEGIN
219      1734      1
220      1735      1      IF .TERM_SIZE EQL 1
221      1736      1      THEN
222      1737      1      BEGIN
223      1738      1      TERM_PTR = RAB [COB$$B_STV0_TERM] ;
224      1739      1      SELECT ONE .RAB [COB$$B_STV0_TERM] OF
225      1740      1      SET
226      1741      1      [ CR,      ! Carriage Return
227      1742      1      TAB ] :      ! Tab
228      1743      1
229      1744      1      BEGIN
230      1745      1      IF .KEY NEQ 0
231      1746      1      THEN
232      1747      1      CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;
233      1748      1      LEGAL = 1 ;
234      1749      1      END ;
235      1750      1
236      1751      1      [ CZ ] :      ! Control z
237      1752      1
238      1753      1      BEGIN
239      1754      1      +
240      1755      1      CONTROL Z hit along with data
241      1756      1      -
242      1757      1      IF (.FLAGS AND V_COB_RPG) NEQ 0
243      1758      1      THEN
244      1759      1      BEGIN      ! Control Z is illegal
245      1760      1      LEGAL = 0 ;      ! for VAX RPG
246      1761      1      TERM_SIZE = 0 ;
247      1762      1      END
248      1763      1      ELSE      ! Special meaning for
249      1764      1      BEGIN      ! VAX COBOL
250      1765      1      COB$$CLEAN UP ( .PARAMETERS, .FLAGS ) ;
251      1766      1      COB$$CONTROL_Z ( .UNIT, .KEY ) ;
252      1767      1      RETURN 0 ;
253      1768      1      END ;
254      1769      1      END ;
255      1770      1
256      1771      1      [ DEL_KEY ] :      ! Delete key
257      1772      1
258      1773      1      BEGIN
259      1774      1      COB$$DELETE_KEY ( .PARAMETERS, .UNIT, .FLAGS ) ;
260      1775      1      NO_BELL = 1 ;      ! Special processing for

```



```

261      M 1776 1      END ;                      ! the DELETE KEY.
262      M 1777 1
263      M 1778 1      [OTHERWISE] :              ! Error - key not a
264      M 1779 1                      ! legal terminator
265      M 1780 1      BEGIN
266      M 1781 1      LEGAL = 0 ;
267      M 1782 1      TERM_SIZE = 0 ;
268      M 1783 1      END ;
269      M 1784 1
270      M 1785 1      END TES ;
271      M 1786 1
272      M 1787 1      ELSE
273      M 1788 1      IF .CHARS_READ EQL 0 AND .RAB [RAB$L_STS] EQL RMSS_EOF
274      M 1789 1      THEN
275      M 1790 1      +
276      M 1791 1      - CONTROL Z hit alone
277      M 1792 1      BEGIN
278      M 1793 1      IF (.FLAGS AND V_COB_RPG) NEQ 0
279      M 1794 1      THEN
280      M 1795 1      BEGIN
281      M 1796 1      LEGAL = 0 ;                      ! Control Z is illegal
282      M 1797 1      TERM_SIZE = 0 ;                ! for VAX RPG
283      M 1798 1      END
284      M 1799 1      ELSE
285      M 1800 1      BEGIN
286      M 1801 1      COB$$CLEAN_UP ( .PARAMETERS, .FLAGS ) ;
287      M 1802 1      COB$$CONTROL_Z ( .UNIT, .KEY ) ;
288      M 1803 1      RETURN 0 ;
289      M 1804 1      END ;
290      M 1805 1      END
291      M 1806 1      ELSE
292      M 1807 1      +
293      M 1808 1      - Escape Sequence as Terminator.
294      M 1809 1      IF .KEY NEQ 0
295      M 1810 1      THEN
296      M 1811 1      +
297      M 1812 1      - COB$$CONTROL_KEY converts terminator sequences to COBOL
298      M 1813 1      - defined sequences and fills in KEY parameter if terminator
299      M 1814 1      - is legal.
300      M 1815 1      BEGIN
301      M 1816 1      IF NOT ( COB$$CONTROL_KEY (TERM_PTR, .TERM_SIZE, .KEY) )
302      M 1817 1      THEN
303      M 1818 1      BEGIN
304      M 1819 1      LEGAL = 0 ;
305      M 1820 1      TERM_SIZE = 0 ;
306      M 1821 1      END
307      M 1822 1      ELSE
308      M 1823 1      LEGAL = 1 ;
309      M 1824 1      END
310      M 1825 1      ELSE
311      M 1826 1      +
312      M 1827 1      - KEY parameter not passed. Escape sequences are not
313      M 1828 1      - legal terminators.
314      M 1829 1
315      M 1830 1
316      M 1831 1
317      M 1832 1
```



```
318 M 1833 1 BEGIN
319 M 1834 1 LEGAL = 0 ;
320 M 1835 1 TERM_SIZE = 0 ;
321 M 1836 1 END ;
322 M 1837 1 END ; ! End $VERIFY_TERMINATOR macro
323 M 1838 1
324 M 1839 1
325 M 1840 1 MACRO
326 M 1841 1 $ERROR_REPROMPT =
327 M 1842 1 !+
328 M 1843 1 Ring terminal bell to signal a CONVERSION error was been made during
329 M 1844 1 data input, restore cursor to original position and perform another
330 M 1845 1 $GET to look for valid data.
331 M 1846 1 !-
332 M 1847 1
333 M 1848 1 BEGIN ! Begin $ERROR_REPROMPT macro
334 M 1849 1
335 M 1850 1 LOCAL
336 M 1851 1 PUT_TOTAL : INITIAL (0), ! # of chars to $PUT
337 M 1852 1 INDEX : INITIAL (0), ! Pointer to RESTORE_CURSOR
338 M 1853 1 RESTORE_CURSOR : VECTOR [5000, BYTE]; ! Holds sequence for restoring
339 M 1854 1 ! cursor position.
340 M 1855 1 COBS$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
341 M 1856 1
342 M 1857 1 !+
343 M 1858 1 If NO ECHO was set no need to do any erasing of input data
344 M 1859 1 !-
345 M 1860 1
346 M 1861 1 IF .YES_NO_ECHO EQL 0
347 M 1862 1 THEN
348 M 1863 1 BEGIN
349 M 1864 1
350 M 1865 1 IF (.PUT_FLAG NEQ 0) AND (.YES_PROTECT EQL 0)
351 M 1866 1 THEN
352 M 1867 1
353 M 1868 1 !+
354 M 1869 1 If no protection set and attributes were turned on - turn them off.
355 M 1870 1 If protection was set, leave the FIELD on the screen. OFF_BUF holds
356 M 1871 1 escape sequence to turn off attributes. OFF_LEN - length of that
357 M 1872 1 sequence.
358 M 1873 1 !-
359 M 1874 1
360 M 1875 1 BEGIN
361 M 1876 1 CH$MOVE ( .OFF_LEN, OFF_BUF [0], RESTORE_CURSOR [0] ) ;
362 M 1877 1 PUT_TOTAL = .OFF_LEN ; ! Total for $PUT so far
363 M 1878 1 END ;
364 M 1879 1
365 M 1880 1 !+
366 M 1881 1 Sequence for reprompting - Backspace, Space, Backspace for each input
367 M 1882 1 character.
368 M 1883 1 !-
369 M 1884 1
370 M 1885 1 INDEX = .PUT_TOTAL ; ! PUT_TOTAL = 0 or .OFF_LEN
371 M 1886 1 INCR P FROM .PUT_TOTAL TO (.PUT_TOTAL+(.CHARS_READ-1)) DO
372 M 1887 1 BEGIN
373 M 1888 1 RESTORE_CURSOR [.INDEX] = BS ; ! Backspace
374 M 1889 1 RESTORE_CURSOR [.INDEX+1] = BLANK ; ! Space
```



```
375 M 1890 1 RESTORE_CURSOR [.INDEX+2] = BS ; ! Backspace
376 M 1891 1 INDEX = .INDEX + 3 ;
377 M 1892 1 END ;
378 M 1893 1 PUT_TOTAL = .PUT_TOTAL + (.CHARS_READ*3) ; ! Total for $PUT so far
379 M 1894 1
380 M 1895 1 IF (.PUT_FLAG NEQ 0) AND (.YES_PROTECT EQL 0)
381 M 1896 1 THEN
382 M 1897 1
383 M 1898 1 !+
384 M 1899 1 ! If no protection set and attributes used - turn them on again.
385 M 1900 1 ! (after deleting all characters from screen). ON_BUF holds escape
386 M 1901 1 ! sequence to turn on attributes. ON_LEN - length of that sequence.
387 M 1902 1 !-
388 M 1903 1
389 M 1904 1 BEGIN
390 M 1905 1 CHSMOVE ( .ON_LEN, ON_BUF [0], RESTORE_CURSOR [.PUT_TOTAL] ) ;
391 M 1906 1 PUT_TOTAL = .PUT_TOTAL + .ON_LEN ; ! Total for $PUT
392 M 1907 1 END ;
393 M 1908 1
394 M 1909 1 END ;
395 M 1910 1
396 M 1911 1 !+
397 M 1912 1 ! Max for $PUT buffer is 1024 (can be increased by changing the max on
398 M 1913 1 ! a SYSGEN parameter). If user input 500 characters the total sequence
399 M 1914 1 ! for reprompting would be 1500 bytes plus possible sequences for turning
400 M 1915 1 ! attributes off and on again, therefore perform a $PUT in sets of 1024
401 M 1916 1 ! until the whole buffer RESTORE_CURSOR has been written to terminal.
402 M 1917 1 !-
403 M 1918 1
404 M 1919 1 BEGIN
405 M 1920 1 LOCAL
406 M 1921 1 P_TOT, ! Length of $PUT.
407 M 1922 1 LAST_WRITE : INITIAL (0) ; ! = 1 for final $PUT -
408 M 1923 1 ! $PUT less than 1024 bytes
409 M 1924 1 WHILE .LAST_WRITE EQL 0 DO
410 M 1925 1 BEGIN
411 M 1926 1 IF .PUT_TOTAL GTR (COBSK_ACC_SIZE - RMS_HEADER) ! COBSK_ACC_SIZE = 1024
412 M 1927 1 THEN
413 M 1928 1 BEGIN ! Need multiple $PUTs.
414 M 1929 1 P_TOT = COBSK_ACC_SIZE - RMS_HEADER ; ! # to Write to screen this time.
415 M 1930 1 PUT_TOTAL = .PUT_TOTAL - .P_TOT ; ! # still to Write.
416 M 1931 1 END
417 M 1932 1 ELSE
418 M 1933 1 BEGIN ! Final $PUT
419 M 1934 1 P_TOT = .PUT_TOTAL ;
420 M 1935 1 LAST_WRITE = 1 ;
421 M 1936 1 END ;
422 M 1937 1 END;
423 M 1938 1
424 M 1939 1 !+
425 M 1940 1 ! Clear screen of invalid input
426 M 1941 1 !-
427 M 1942 1
428 M 1943 1 COB$$RMS_PUT_BUFFER ( RESTORE_CURSOR [0], .P_TOT, .FLAGS ) ;
429 M 1944 1 END ;
430 M 1945 1
431 M 1946 1 !+
```



```
432      ! Perform another $GET - looking for valid input
433      !-
434      M 1948 1
435      M 1949 1
436      M 1950 1      RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
437      M 1951 1      COB$$RMS_GET ( .RAB, .FUNC_VAL, .ACC_SIZE, .PUT_HERE [DSC$A_POINTER] ) ;
438      M 1952 1
439      M 1953 1      REPROMPT_DONE = 1 ;          ! Signal that REPROMPT has been
440      M 1954 1          done.
441      M 1955 1      END ;          ! End of $ERROR_REPROMPT macro
442      M 1956 1      % ;
443      M 1957 1
444      M 1958 1      MACRO
445      M 1959 1      $BIND_PARAMETERS =
446      M 1960 1      +
447      M 1961 1      Put data used by many of the subroutines in a vector of data.
448      M 1962 1      BIND all the separate names that can be used to identify the
449      M 1963 1      various elements of the vector.
450      M 1964 1      -
451      M 1965 1      BIND
452      M 1966 1      PUT_HERE      = PARAMETERS [0] : BLOCK [ ,BYTE], ! Buffer to hold input
453      M 1967 1      NEXT_CHAR     = PARAMETERS [3] : VECTOR [ ,BYTE], ! Buffer used for
454      M 1968 1          PROTECTION check
455      M 1969 1      ACC_SIZE      = PARAMETERS [6] : WORD, ! Length for RMS $GET
456      M 1970 1      CHARS_READ    = PARAMETERS [7], ! Number of input characters
457      M 1971 1      FUNC_VAL      = PARAMETERS [8], ! QIO Function Modifiers used
458      M 1972 1          in the item list for RMS $GET
459      M 1973 1      TERM_SIZE     = PARAMETERS [9], ! Size of terminator
460      M 1974 1      TERM_LOC      = PARAMETERS [10], ! Location of terminator
461      M 1975 1      TERM_PTR      = PARAMETERS [11], ! Pointer to terminator in buffer
462      M 1976 1      TERM_IN_NEXT  = PARAMETERS [12], ! = 1 if terminator in NEXT_CHAR
463      M 1977 1      TERM_FROM_DEL = PARAMETERS [13], ! Flag from COB$$DELETE_KEY to
464      M 1978 1          COB$$ILLEGAL_TERM
465      M 1979 1      LEGAL         = PARAMETERS [14], ! = 0 if illegal terminator hit
466      M 1980 1      YES_PROTECT   = PARAMETERS [15], ! = 1 if PROTECTED requested
467      M 1981 1      YES_DEFAULT   = PARAMETERS [16], ! = 1 if DEFAULT used as input
468      M 1982 1      PUT_FLAG      = PARAMETERS [17], ! Flag for turning on attributes
469      M 1983 1      OFF_BUF       = PARAMETERS [18] : VECTOR [ ,BYTE], ! Holds esc seq to
470      M 1984 1          ! turn off attributes
471      M 1985 1      OFF_LEN       = PARAMETERS [21] ; ! Length of esc seq in OFF_BUF
472      M 1986 1      % ;
473      M 1987 1      !-
474      M 1988 1      The following tables convert the UNIT number into a logical name.
475      M 1989 1      !-
476      M 1990 1      MACRO
477      M 1991 1      DESC_(A) = UPLIT BYTE(%ASCIC A) - BASE %;
478      M 1992 1      BIND
479      M 1993 1      BASE = UPLIT(REP 0 OF (0)),
480      M 1994 1      COB_TABLE = UPLIT(
481      M 1995 1          DESC_('COB$INPUT'),
482      M 1996 1          DESC_('COB$OUTPUT'),
483      M 1997 1          DESC_('COB$CONSOLE'),
484      M 1998 1          DESC_('COB$CARDREADER'),
485      M 1999 1          DESC_('COB$PAPERTAPEREREADER'),
486      M 2000 1          DESC_('COB$LINEPRINTER'),
487      M 2001 1          DESC_('COB$PAPERTAPEPUNCH')) : VECTOR[ NUM_UNITS ],
488      M 2002 1      SYS_TABLE = UPLIT(
489      M 2003 1          DESC_('SYSS$INPUT'),
```



```
489 2004 1 DESC ('SYSS$OUTPUT'),
490 2005 1 DESC ('SYSS$ERROR'),
491 2006 1 DESC ('SYSS$INPUT'),
492 2007 1 DESC ('SYSS$INPUT'),
493 2008 1 DESC ('SYSS$OUTPUT'),
494 2009 1 DESC ('SYSS$OUTPUT'),
495 2010 1 DESC ('SYSS$OUTPUT');
496 2011 1
497 2012 1 ! EXTERNAL REFERENCES:
498 2013 1 !
499 2014 1 EXTERNAL ROUTINE
500 2015 1
501 2016 1 COB$$CONTROL KEY,
502 2017 1 COB$$ACC_CONVERT,
503 2018 1 COB$$OPEN_OUT : NOVALUE,
504 2019 1 LIB$STOP : NOVALUE,
505 2020 1 LIB$GET_VM,
506 2021 1 LIB$FREE_VM,
507 2022 1 STR$GET1_DX,
508 2023 1 STR$DUPL_CHAR,
509 2024 1 STR$FREE1_DX,
510 2025 1 STR$COPY_R,
511 2026 1 COB$$SETUP_TERM_TYPE,
512 2027 1 COB$$SET_ATTRIBUTES_ONLY ;
513 2028 1
514 2029 1 EXTERNAL LITERAL
515 2030 1 COB$_ERRDURACC,
516 2031 1 COB$_FAIGET_VM,
517 2032 1 COB$_EOFON_ACC,
518 2033 1 COB$_INVDEFVAL,
519 2034 1 COB$_INVARG ;
520 2035 1
521 2036 1 EXTERNAL
522 2037 1 COB$$AL_WRITE_RAB : VECTOR,
523 2038 1 COB$$AW_WRITE_IFI : VECTOR [,WORD],
524 2039 1 COB$$AB_USPCODE : VECTOR [,BYTE],
525 2040 1 COB$$AB_PREV : VECTOR [,BYTE],
526 2041 1 COB$$ACC_TERM_TYPE,
527 2042 1 COB$TERM_TYPE;
```

VECTOR[ NUM\_UNITS];

! Translate terminator  
! Conversion routine  
! Open for output  
! Signals fatal error  
! Get virtual memory  
! Free virtual memory  
! Allocate a string  
! Duplicate character n times  
! Deallocate a string  
! Copy a string by ref  
! Setup terminal type  
! Set bold, reverse, blink,  
! underline

! Error during DISPLAY  
! Failure to get VM  
! EOF on ACCEPT  
! DEFAULT value too large  
! Invalid Argument(s)

! Address of RAB  
! Internal file identifiers  
! Prefix and Post upspacing  
! History of previous call  
! Terminal type for ACCEPT  
! Terminal type for DISPLAY



```

529 2043 1 %SBTTL 'COBSACCEPT - Version 1 ACCEPT Statement'
530 2044 1 GLOBAL ROUTINE COBSACCEPT (UNIT, STRING) =
531 2045 1
532 2046 1 !++
533 2047 1 FUNCTIONAL DESCRIPTION:
534 2048 1
535 2049 1 Reads a record from specified unit and delivers record to
536 2050 1 caller's string.
537 2051 1
538 2052 1 FORMAL PARAMETERS:
539 2053 1
540 2054 1 UNIT.rbu.va Byte integer unit number designating the unit from
541 2055 1 which the string is to be read, followed by byte
542 2056 1 flag indicating whether routine should (false) abort,
543 2057 1 or (true) return status on RMSS_EOF.
544 2058 1
545 2059 1 STRING.wt.ds The address of a fixed-string descriptor to
546 2060 1 receive the string read.
547 2061 1
548 2062 1 IMPLICIT INPUTS:
549 2063 1
550 2064 1 Status of whether the file in question is currently open.
551 2065 1
552 2066 1 IMPLICIT OUTPUTS:
553 2067 1
554 2068 1 Updated status of the file just used.
555 2069 1
556 2070 1 ROUTINE VALUE:
557 2071 1
558 2072 1 If .UNIT[1] is false:
559 2073 1 Unspecified.
560 2074 1
561 2075 1 If .UNIT[1] is true:
562 2076 1 Either true or false, indicating success or EOF, respectively.
563 2077 1
564 2078 1 SIDE EFFECTS:
565 2079 1
566 2080 1 Reads a record from the designated unit.
567 2081 1
568 2082 1 --
569 2083 1
570 2084 2 BEGIN
571 2085 2 MAP
572 2086 2 UNIT: VECTOR[BYTE],
573 2087 2 STRING: REF BLOCK[8, BYTE];
574 2088 2
575 2089 2 LOCAL
576 2090 2 RAB: REF $RAB_DECL,
577 2091 2 STATUS,
578 2092 2 DESCR: BLOCK [8, BYTE],
579 2093 2 TERM_SIZE,
580 2094 2 BUFFER: VECTOR [COBSK_ACC_SIZE, BYTE];
581 2095 2
582 2096 2 IF .UNIT[0] GTRU COBSK_UNIT_MAX
583 2097 2 THEN
584 2098 2 LIB$STOP(COBS_INVARG);
585 2099 2
```



```
586 2100 2 ! If this file is not open, open it.
587 2101 2 ! 0 as second parameter to COB$$OPEN_IN signifies that VAX COBOL is used.
588 2102 2
589 2103 2 IF .COB$$AL_WRITE_RAB[.UNIT[0]] EQL 0
590 2104 2 THEN
591 2105 2     COB$$OPEN_IN(.UNIT[0], 0);
592 2106 2
593 2107 2 ! Perform a Linefeed when a Version 1 ACCEPT statement follows a
594 2108 2 ! Version 3 ACCEPT statement with advancing.
595 2109 2
596 2110 2 IF .COB$$AB_PREV [0] EQL ACC_ADV
597 2111 2 THEN COB$$RMS_PUT_BYTE ( LINE_FD, 0 );
598 2112 2
599 2113 2 ! Read a record into our buffer or caller's buffer, whichever is larger.
600 2114 2
601 2115 2 RAB = .COB$$AL_WRITE_RAB[.UNIT[0]];
602 2116 2 IF .STRING[DSC$W_LENGTH] GTRU COB$K_ACC_SIZE
603 2117 2 THEN
604 2118 2     BEGIN
605 2119 2     RAB[RAB$W_USZ] = .STRING[DSC$W_LENGTH];
606 2120 2     RAB[RAB$L_UBF] = .STRING[DSC$A_POINTER];
607 2121 2     END
608 2122 2 ELSE
609 2123 2     BEGIN
610 2124 2     RAB[RAB$W_USZ] = COB$K_ACC_SIZE;
611 2125 2     RAB[RAB$L_UBF] = BUFFER;
612 2126 2     END;
613 2127 2
614 2128 2 !+
615 2129 2 ! Turn off RAB [RAB$V_ETO] just in case a 'screen enhancement ACCEPT'
616 2130 2 ! was performed before this one. COB$$RMS_GET set RAB [RAB$V_ETO]
617 2131 2 ! to signal to RMS to expect an 'extended terminal' $GET. Here we
618 2132 2 ! only want a simple $GET with no bells and whistles. If RAB [RAB$V_ETO]
619 2133 2 ! is not turned off unwanted behavior will result.
620 2134 2 !-
621 2135 2
622 2136 2 RAB [RAB$V_ETO] = 0 ;
623 2137 2
624 2138 2 ! Read the record.
625 2139 2
626 2140 2 WHILE $GET(RAB = .RAB) EQL RMS$_RSA DO $WAIT(RAB = .RAB);
627 2141 2
628 2142 2
629 2143 2 IF NOT .RAB[RAB$L_STS]
630 2144 2 THEN
631 2145 2     LIB$STOP(
632 2146 2     (IF .RAB[RAB$L_STS] EQL RMS$_EOF
633 2147 2     THEN
634 2148 2         IF .UNIT[1]
635 2149 2         THEN RETURN 0
636 2150 2         ELSE COB$ EOFON_ACC
637 2151 2         ELSE COB$ ERRDURACCT,
638 2152 2         1, .RAB+RAB$C_BLN, .RAB[RAB$L_STS], .RAB[RAB$L_STV]);
639 2153 2
640 2154 2 !+
641 2155 2 ! Check if the terminator size is greater than 1. If it is,
642 2156 2 ! this indicates that the terminator string is an escape sequence.
```



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COBSACCEPT - Version 1 ACCEPT Statement

L 2  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 13  
(3)

```

: 643      2157 2 ! Return the entire escape sequence in the user's buffer.
: 644      2158 2 !-
: 645      2159 2
: 646      2160 2     TERM_SIZE = .RAB[RAB$W_STV2];           ! Get terminator size
: 647      2161 2
: 648      2162 2     CH$COPY( (IF .TERM SIZE GTR 1
: 649      2163 2         THEN .RAB[RAB$W_RSZ] + .TERM_SIZE
: 650      2164 2         ELSE .RAB[RAB$W_RSZ] ),
: 651      2165 2         .RAB[RAB$L_UBF], %C'-', .STRING[DSC$W_LENGTH], .STRING[DSC$A_POINTER]);
: 652      2166 2
: 653      2167 2     !+
: 654      2168 2     ! VAX COBOL Version 1 / Version 3 interaction.
: 655      2169 2     ! Interaction with COB$ACC_SCR - Perform a Carriage Return if necessary
: 656      2170 2     ! and signal that this is an ACCEPT with advancing.
: 657      2171 2     !-
: 658      2172 2
: 659      2173 2     IF .ACC_SCR
: 660      2174 2     THEN
: 661      2175 2         COB$$RMS PUT BYTE ( CARR_RET, 0 ) ;
: 662      2176 2         COB$$AB_PREV[0] = ACC_ADV ;
: 663      2177 2
: 664      2178 2     RETURN 1;
: 665      2179 1     END;                                     ! End COBSACCEPT
```

.TITLE COBSACCEPT COBSACCEPT - VAX COBOL ACCEPT Statem  
ent

.IDENT \1-018\

.PSECT \_COB\$DATA,NOEXE, PIC,2

```

00000000 00000 ACC_SCR::
                                .LONG 0
                                00004 XABTRM: .BLKB 36
                                00028 XAB_ITMLST:
                                .BLKB 28
OD F5 FF F6 00044 MASK_VECTOR:
                                .BYTE -10, -1, -11, 13
                                00# 00048 .BYTE 0[11]
                                80 00053 .BYTE -128
                                00 00054 .BYTE 0
                                80 00055 .BYTE -128
                                00 00056 .BYTE 0
                                08 00057 .BYTE 8
```

.PSECT \_COB\$CODE,NOWRT, SHR, PIC,2

```

52 45 44 41 45 54 55 50 4E 49 24 42 4F 43 09 00000 P.AAA: .BLKB 0
52 45 50 41 54 52 44 52 41 43 24 42 4F 43 0A 00000 P.AAC: .ASCII <9>\COB$INPUT\
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 0B 0000A P.AAD: .ASCII <10>\COB$OUTPUT\
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 0C 00015 P.AAE: .ASCII <11>\COB$CONSOLE\
52 45 50 41 54 52 45 50 41 50 24 42 4F 43 0D 00021 P.AAF: .ASCII <14>\COB$CARDREADER\
52 45 50 41 54 52 45 50 41 50 24 42 4F 43 0E 00030 P.AAG: .ASCII <19>\COB$PAPERTAPEREREADER\
45 54 4E 49 52 50 45 4E 49 4C 24 42 4F 43 0F 00044 P.AAH: .ASCII <15>\COB$LINEPRINTER\
50 45 50 41 54 52 45 50 41 50 24 42 4F 43 10 00053 P.AAI: .ASCII <18>\COB$PAPERTAPEPUNCH\
52 45 50 41 54 52 45 50 41 50 24 42 4F 43 11 00054
```



Address	Instruction	Comment	Hex	Symbol
57	00000000G	00 00FC 00000	00 9E 00002	.ENTRY COB\$ACCEPT, Save R2,R3,R4,R5,R6,R7
56	00000000G	00 9E 00009	00 9E 00009	MOVAB COB\$\$AB_PREV, R7
5E	FBF8	CE 9E 00010	CE 9E 00010	MOVAB LIB\$STOP, R6
52	04	AC 9A 00015	AC 9A 00015	MOVAB -1032(SP), SP
06		52 91 00019	52 91 00019	MOVZBL UNIT, R2
		09 1B 0001C	09 1B 0001C	CMPB R2, #6
	00000000G	8F DD 0001E	8F DD 0001E	BLEQU 1\$
66		01 FB 00024	01 FB 00024	PUSHL #COB\$ INVARG
53	00000000G	0042 DE 00027	0042 DE 00027	CALLS #1, LIB\$STOP
		63 D5 0002F	63 D5 0002F	MOVAL COB\$\$AL_WRITE_RAB[R2], R3
		09 12 00031	09 12 00031	TSTL (R3)
		7E D4 00033	7E D4 00033	BNEQ 2\$
		52 DD 00035	52 DD 00035	CLRL -(SP)
0000V	CF	02 FB 00037	02 FB 00037	PUSHL R2
	04	67 91 0003C	67 91 0003C	CALLS #2, COB\$\$OPEN_IN
		08 12 0003F	08 12 0003F	CMPB COB\$\$AB_PREV, #4
		01 7D 00041	01 7D 00041	BNEQ 3\$
0000V	7E	02 FB 00044	02 FB 00044	MOVQ #1, -(SP)
	CF	63 D0 00049	63 D0 00049	CALLS #2, COB\$\$RMS_PUT_BYTE
	52	AC D0 0004C	AC D0 0004C	MOVL (R3), RAB
	53	B1 00050	B1 00050	MOVL STRING, R3
0400	8F			CMPW (R3), #1024



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$ACCEPT - Version 1 ACCEPT Statement

N 2  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 15  
(3)

	20	A2		0B	1B	00055	BLEQU	4\$			
	24	A2	04	63	B0	00057	MOVW	(R3), 32(RAB)			2119
				A3	D0	00058	MOVL	4(R3), 36(RAB)			2120
				0A	11	00060	BRB	5\$			2116
	20	A2	0400	8F	B0	00062	MOVW	#1024, 32(RAB)			2124
	24	A2		6E	9E	00068	MOVAB	BUFFER, 36(RAB)			2125
	07	A2		10	8A	0006C	BICB2	#16, 7(RAB)			2136
				52	DD	00070	PUSHL	RAB			2140
00000000G		00		01	FB	00072	CALLS	#1, SYSSGET			
000182DA		8F		50	D1	00079	CMPL	R0, #99034			
				0B	12	00080	BNEQ	7\$			
				52	DD	00082	PUSHL	RAB			
00000000G		00		01	FB	00084	CALLS	#1, SYSSWAIT			
	2B		08	E3	11	0008B	BRB	6\$			
	7E		08	A2	E8	0008D	BLBS	8(RAB), 10\$			2143
			44	A2	7D	00091	MOVQ	8(RAB), -(SP)			2152
				A2	9F	00095	PUSHAB	68(RAB)			
				01	DD	00098	PUSHL	#1			2145
0001827A		8F	08	A2	D1	0009A	CMPL	8(RAB), #98938			2146
	47		05	0F	12	000A2	BNEQ	8\$			
	50	00000000G		AC	E8	000A4	BLBS	UNIT+1, 14\$			2148
				8F	D0	000A8	MOVL	#COB\$_EOFON_ACC, R0			
				50	DD	000AF	PUSHL	R0			
				06	11	000B1	BRB	9\$			
			00000000G	8F	DD	000B3	PUSHL	#COB\$_ERRDURACC			2146
	66			05	FB	000B9	CALLS	#5, LIB\$STOP			
	50		0E	A2	3C	000BC	MOVZWL	14(RAB), TERM_SIZE			2160
	01			50	D1	000C0	CMPL	TERM_SIZE, #1			2162
				09	15	000C3	BLEQ	11\$			
	51		22	A2	3C	000C5	MOVZWL	34(RAB), R1			2163
	50			51	C0	000C9	ADDL2	R1, R0			
				04	11	000CC	BRB	12\$			
				A2	3C	000CE	MOVZWL	34(RAB), R0			2164
63	20	24	22	50	2C	000D2	MOVCS	R0, @36(RAB), #32, (R3), @4(R3)			2165
			04	B3		000D8					
				EF	E9	000DA	BLBC	ACC SCR, 13\$			2173
				7E	7C	000E1	CLRQ	-(SP)			2175
0000V	CF			02	FB	000E3	CALLS	#2, COB\$\$RMS_PUT_BYTE			
	67			04	90	000E8	MOVB	#4, COB\$\$AB_PREV			2176
	50			01	D0	000EB	MOVL	#1, R0			2178
					04	000EE	RET				
				50	D4	000EF	CLRL	R0			2179
				04	000F1		RET				

; Routine Size: 242 bytes, Routine Base: \_COB\$CODE + 00EC



```

: 667 2180 1 %SBTTL 'COBSACC_SCR - ACCEPT with screen enhancements'
: 668 2181 1 GLOBAL ROUTINE COBSACC_SCR ( UNIT      : VECTOR [2,BYTE],
: 669 2182 1      STRING_DEST : REF $STR$DESCRIPTOR,
: 670 2183 1      FLAGS,      : REF $STR$DESCRIPTOR,
: 671 2184 1      DEFAULT    : REF $STR$DESCRIPTOR,
: 672 2185 1      SIZE,      : REF $STR$DESCRIPTOR,
: 673 2186 1      KEY        : REF $STR$DESCRIPTOR,
: 674 2187 1      LENGTH    : REF $STR$DESCRIPTOR,
: 675 2188 1      ) =
: 676 2189 1
: 677 2190 1 ++
: 678 2191 1 FUNCTIONAL DESCRIPTION:
: 679 2192 1
: 680 2193 1     Performs COBOL ACCEPT statement with screen enhancements.
: 681 2194 1     Reads a record from a specified UNIT and deposits record in
: 682 2195 1     STRING_DEST.
: 683 2196 1     A call to COBSPOS_ACCEPT is made by the VAX COBOL Compiler
: 684 2197 1     prior to the call to COBSACC_SCR to set cursor position and
: 685 2198 1     perform any screen or line erasing.
: 686 2199 1
: 687 2200 1 CALLING SEQUENCE:
: 688 2201 1
: 689 2202 1     RETURN_STATUS.wlc.v = COBSACC_SCR ( UNIT.rbu.va, STRING_DEST.mt.ds,
: 690 2203 1                                     [FLAGS.rlu.v], [DEFAULT.rt.dx],
: 691 2204 1                                     [SIZE.rlu.v], [KEY.wt.ds],
: 692 2205 1                                     [LENGTH.wlu.r] )
: 693 2206 1
: 694 2207 1 FORMAL PARAMETERS:
: 695 2208 1
: 696 2209 1     UNIT.rbu.va      Array of two unsigned byte integers.
: 697 2210 1                 The first byte is the unit number designating the
: 698 2211 1                 device from which the string is to be read.
: 699 2212 1                 The second byte indicates whether the routine should
: 700 2213 1                 abort or return to the calling program.
: 701 2214 1                 Byte 2 = 0 - routine will abort on control z
: 702 2215 1                         and reprompt on conversion errors.
: 703 2216 1                         = 1 - ( AT END )
: 704 2217 1                         routine will return to calling program
: 705 2218 1                         on control z and reprompt on conversion
: 706 2219 1                         errors.
: 707 2220 1                         = 2 - ( ON EXCEPTION )
: 708 2221 1                         routine will return to calling program
: 709 2222 1                         on control z and conversion errors.
: 710 2223 1
: 711 2224 1     STRING_DEST.mt.ds Address of descriptor to receive the read input.
: 712 2225 1
: 713 2226 1     FLAGS.rlu.v      Screen enhancement flag:
: 714 2227 1
: 715 2228 1                     bit 0 - bold
: 716 2229 1                     bit 1 - reverse
: 717 2230 1                     bit 2 - blink
: 718 2231 1                     bit 3 - underline
: 719 2232 1                     bit 4 - bell
: 720 2233 1                     bit 5 - conversion
: 721 2234 1                     bit 6 - decimal point is comma
: 722 2235 1                     bit 7 - 0 to allow space for sign in PROTECTED
: 723 2236 1                     ACCEPT, 1 no allowance for sign
:                          bit 8 - protect
```



```

724 2237 1 bit 9 - no-echo
725 2238 1 bit 10 - 0 advancing, 1 no advancing
726 2239 1 bit 11 - 0 for VAX COBOL, 1 for VAX RPG
727 2240 1
728 2241 1 DEFAULT.rt.dx Default source moved to destination descriptor
729 2242 1 (STRING_DEST) in the event of null input.
730 2243 1
731 2244 1 SIZE.rlu.v Size of protected field. Only applicable if the
732 2245 1 protected flag is set.
733 2246 1
734 2247 1 KEY.wt.ds Destination of the receiving field of the control key
735 2248 1
736 2249 1 LENGTH.wlu.r Destination of the number of characters read
737 2250 1
738 2251 1 IMPLICIT INPUTS:
739 2252 1
740 2253 1 Status of whether the input file is currently open.
741 2254 1
742 2255 1 IMPLICIT OUTPUTS:
743 2256 1
744 2257 1 Updated status of file
745 2258 1
746 2259 1 ROUTINE VALUE:
747 2260 1
748 2261 1 If .UNIT[1] is false : Unspecified.
749 2262 1 If .UNIT[1] is true : Either true or false, indicating success or
750 2263 1 EOF, respectively.
751 2264 1
752 2265 1 SIDE EFFECTS:
753 2266 1
754 2267 1 Reads a record from a designated uint.
755 2268 1
756 2269 1 --
757 2270 1
758 2271 2 BEGIN
759 2272 2
760 2273 2 LOCAL
761 2274 2
762 2275 2 Note; other declarations are in the macro $BIND_PARAMETERS.
763 2276 2
764 2277 2 RAB : REF $RAB_DECL,
765 2278 2 PUT_SIZE : WORD,
766 2279 2
767 2280 2 ON_BUF : VECTOR [20,BYTE],
768 2281 2
769 2282 2 ON_LEN : INITIAL (0),
770 2283 2 FUNC_VAL_2,
771 2284 2
772 2285 2 PROT_OK : INITIAL (0),
773 2286 2 CONV_OK : INITIAL (0),
774 2287 2 REPRMPT_DONE : INITIAL (0),
775 2288 2
776 2289 2 YES_CONV : INITIAL (0),
777 2290 2 YES_NO_ECHO : INITIAL (0),
778 2291 2 YES_SIGN : INITIAL (1),
779 2292 2
780 2293 2 P_DATA_TYPE : INITIAL (0),
```

ACC\_SIZE plus 5 (for escape sequences)  
Holds escape seq to turn on terminal attributes  
Length of ON\_BUF  
QIO Function Modifiers used in the item list for RMS \$GET  
= 1 if no Protection errors  
= 1 if no Conversion errors  
= 1 if reprompt performed in response to a Conversion error  
= 1 if Conversion requested  
= 1 if No-Echo requested  
= 1 if no allowance for sign given. NOTE - initialized to 1  
PP99 or 99PP data types



```

781 2294 2      ZEROES,          ! %X'0' filler
782 2295 2      BLANKS,         ! %C' ' filler
783 2296 2      PARAMETERS      : VECTOR [22]      ! Buffer to hold data to be
784 2297 2      INITIAL (REP 22 OF (0)) ; ! passed to subroutines
785 2298 2      BUILTIN
786 2299 2      NULLPARAMETER ;
787 2300 2
788 2301 2      LITERAL
789 2302 2      F_PROT_SIZE = 13,          ! # of chars allowed for input
790 2303 2      D_PROT_SIZE = 22 ;        ! when PROTECTED is requested
791 2304 2      !+                                     ! for floating and double fl.
792 2305 2      Bind PARAMETERS to other names.
793 2306 2      !-
794 2307 2      $BIND_PARAMETERS ;
795 2308 2
796 2309 2      !+
797 2310 2      Fillers - used by STR$DUPL_CHAR, therefore they cannot be literals
798 2311 2      !-
799 2312 2      ZEROES = %X'0' ;
800 2313 2      BLANKS = %C' ' ;
801 2314 2
802 2315 2      !+
803 2316 2      Put ACCEPTed data from RMS $GET in this buffer.
804 2317 2      !-
805 2318 2
806 2319 2      PUT_HERE [DSC$W_LENGTH] = 0 ;
807 2320 2      PUT_HERE [DSC$B_DTYPE] = DSC$K_DTYPE_NL ;
808 2321 2      PUT_HERE [DSC$B_CLASS] = DSC$K_CLASS_D ;
809 2322 2      PUT_HERE [DSC$A_POINTER] = 0 ;
810 2323 2
811 2324 2      !+
812 2325 2      Determine if PROTECTION has been requested.
813 2326 2      If so, set the size of the field by either the value of the SIZE parameter
814 2327 2      or the length field of the STRING DEST descriptor.
815 2328 2      If no PROTECTION requested, use COBSK_ACC_SIZE (1024 - same as
816 2329 2      V1 Accept).
817 2330 2      Also make adjustments if both PROTECTION and CONVERSION are requested -
818 2331 2      add room for sign and a decimal point, in some cases look at DSC$B_DIGITS
819 2332 2      instead of DSC$W_LENGTH.
820 2333 2      'P' data types need special handling.
821 2334 2      Use STR$GET1_DX to allocate space for dynamic string PUT_HERE.
822 2335 2      !-
823 2336 2
824 2337 2      IF ( .FLAGS AND V_CONV ) NEQ 0 THEN YES_CONV = 1 ; ! Avoid BLISS
825 2338 2      IF ( .FLAGS AND V_NO_SIGN ) NEQ 0 THEN YES_SIGN = 0 ; ! optimization problems
826 2339 2
827 2340 2      IF ( .FLAGS AND V_PROTECT ) NEQ 0
828 2341 2      THEN
829 2342 2          BEGIN ! Begin Protect Size
830 2343 2          YES_PROTECT = 1 ;
831 2344 2          IF .SIZE NEQ 0
832 2345 2          THEN ACC_SIZE = .SIZE ! Use SIZE
833 2346 2          ELSE
834 2347 2              BEGIN ! Begin no SIZE param
835 2348 2              LOCAL
836 2349 2              pp99 : initial (0) ; ! Scale for PP99 data
837 2350 2              ! type
```



```

838 2351 4
839 2352 4      pp99 = .string_dest [dsc$b_digits] + .string_dest [dsc$b_scale] ;
840 2353 4      ACC_SIZE = .STRING_DEST [DSC$W_LENGTH] ;      ! Use STRING_DEST
841 2354 4
842 2355 4
843 2356 4      !+
844 2357 4      Special case "P" data types (each "P" specifies an assumed scaling position).
845 2358 4      NOTE: All code pertaining to the "P" data type is in lowercase. Since "P"
846 2359 4      data types are such an off the wall issue, leaving this code in lowercase is
847 2360 4      the best way to avoid "P" code interfering with "normal" data types.
848 2361 6      !-
849 2362 7      if ((.string_dest [dsc$b_class] eql dsc$k_class_sd )
850 2363 5          and ((.pp99 lss 0)
851 2364 4              or (.string_dest[dsc$b_scale] gtr 0)))
852 2365 5          ! P Picture of PP99
853 2366 5          ! P Picture of 99PP.
854 2367 5      then
855 2368 5          begin
856 2369 5              ! begin P data types
857 2370 5              p_data_type = 1 ;
858 2371 5              if .pp99 lss 0
859 2372 5                  ! P Picture of PP99
860 2373 5              then
861 2374 5                  acc_size = abs (.string_dest[dsc$b_scale])
862 2375 6              else
863 2376 6                  acc_size = .pp99 ;
864 2377 6              if .yes_conv
865 2378 6              then
866 2379 6                  begin
867 2380 6                      !+
868 2381 6                      Allow space for a decimal point for PP99 but not 99pp.
869 2382 6                      !-
870 2383 6                      if .pp99 lss 0
871 2384 6                      then
872 2385 6                          acc_size = .acc_size + 1 ;
873 2386 6                          ! decimal point for pp99
874 2387 6                      !+
875 2388 6                      Because we are reading the digits and scale fields,
876 2389 6                      all numeric data types will need an extra space for
877 2390 6                      the sign - except Numeric Unsigned.
878 2391 6                      !-
879 2392 6                      if .string_dest [dsc$b_dtype] neq dsc$k_dtype_nu
880 2393 6                      then
881 2394 6                          acc_size = .acc_size + 1 ;
882 2395 6                      !+
883 2396 6                      Additional check for VAX_11 COBOL COMP and COMP3
884 2397 8                      data types - if YES_SIGN= 0 then do not include
885 2398 8                      space for sign.
886 2399 8                      !-
887 2400 8                      if (((.string_dest [dsc$b_dtype] eql dsc$k_dtype_w ) or
888 2401 8                          (.string_dest [dsc$b_dtype] eql dsc$k_dtype_wu ) or
889 2402 8                          (.string_dest [dsc$b_dtype] eql dsc$k_dtype_l ) or
890 2403 8                          (.string_dest [dsc$b_dtype] eql dsc$k_dtype_lu ) or
891 2404 7                          (.string_dest [dsc$b_dtype] eql dsc$k_dtype_q ) or
892 2405 6                          (.string_dest [dsc$b_dtype] eql dsc$k_dtype_qu ) or
893 2406 6                          (.string_dest [dsc$b_dtype] eql dsc$k_dtype_p ))
894 2407 5                          and .yes_sign eql 0 )
                        then
                        acc_size = .acc_size - 1 ;
                        end ;
```



```

895      2408 5      end                                ! end P data types
896      2409 5
897      2410 4
898      2411 4
899      2412 4
900      2413 4
901      2414 5
902      2415 5      else
903      2416 5          !+ Non P data type
904      2417 5          !-
905      2418 5          begin                                ! Begin non P data type
906      2419 5          IF .YES_CONV                        ! Adjust ACC_SIZE
907      2420 5          THEN
908      2421 5              !+
909      2422 6              Make room for overpunch sign.
910      2423 7              Packed data type - check to see if sign should be
911      2424 7              included.
912      2425 8              !-
913      2426 7              BEGIN
914      2427 6              IF ((.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_NRO ) OR
915      2428 6                  (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_NLO ) OR
916      2429 6                  (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_P AND
917      2430 6                      .YES_SIGN ))
918      2431 6              THEN
919      2432 6                  ACC_SIZE = .ACC_SIZE + 1 ;
920      2433 6              !+
921      2434 6                  COMP - look at digits field plus one for sign, only if
922      2435 6                  conversion is requested.
923      2436 6                  VAX_COBOL always sends an SD descriptor for W, L, Q when
924      2437 6                  conversion is used.
925      2438 7                  Check to see if sign should be included.
926      2439 6                  !-
927      2440 8              IF (.STRING_DEST [DSC$B_CLASS] EQL DSC$K_CLASS_SD )
928      2441 8              THEN
929      2442 8                  IF (((.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_W ) OR
930      2443 8                      (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_WU ) OR
931      2444 8                      (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_L ) OR
932      2445 8                      (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_LU ) OR
933      2446 7                      (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_Q ) OR
934      2447 6                      (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_QU ))
935      2448 6                      AND .YES_SIGN )
936      2449 6              THEN
937      2450 6                  ACC_SIZE = .STRING_DEST [DSC$B_DIGITS] + 1 ;
938      2451 6              !+
939      2452 6                  Floating pt - 13 for Floating, 22 for Double Floating.
940      2453 6              !-
941      2454 7              IF (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_F )
942      2455 6              THEN
943      2456 6                  ACC_SIZE = F_PROT_SIZE ;
944      2457 7              IF (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_D )
945      2458 6              THEN
946      2459 6                  ACC_SIZE = D_PROT_SIZE ;
947      2460 6              !+
948      2461 6                  Make room for decimal point
949      2462 6              !-
950      2463 6
951      2464 6
```



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COBSACC\_SCR - ACCEPT with screen enhancements

G 3  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 21  
(4)

```

952 2465 7          IF (.STRING_DEST [DSC$B_CLASS] EQL DSC$K_CLASS_SD )
953 2466 6          THEN
954 2467 6              ACC_SIZE = .ACC_SIZE + 1 ;
955 2468 5              END ;
956 2469 4              end ;
957 2470 3          END ;
958 2471 3              ! End non P data type
959 2472 2          ELSE
960 2473 2              ACC_SIZE = COB$K_ACC_SIZE - RMS_HEADER ;
961 2474 2              ! 1024 - 14 is same
962 2475 2              ! limit as a DISPLAY
963 2476 2          !+
964 2477 2          Allocate enough room in PUT_HERE to hold the terminator escape sequences.
965 2478 2          Most sequences are 4 bytes or less. (PUT_SIZE is 5 more than ACC_SIZE.)
966 2479 2          Note: PUT_SIZE used, not ACC_SIZE.
967 2480 2          !-
968 2481 2          PUT_SIZE = .ACC_SIZE ;
969 2482 2          IF .ACC_SIZE LSS 920 THEN PUT_SIZE = .ACC_SIZE + 5 ;
970 2483 2
971 2484 2          IF NOT ( STR$GET1 DX ( %REF (.PUT_SIZE ), PUT_HERE ))
972 2485 2          THEN LIB$STOP ( COB$_ERRDURACC ) ;
973 2486 2
974 2487 2          !+
975 2488 2          Check first byte of UNIT param.
976 2489 2          If this file is not open, open it. (Note: only first byte of UNIT is
977 2490 2          sent to COB$$OPEN_IN)
978 2491 2          !-
979 2492 2
980 2493 2          IF .UNIT[0] GTRU COB$K_UNIT_MAX
981 2494 2          THEN
982 2495 2              LIB$STOP ( COB$_INVARG ) ;
983 2496 2
984 2497 2          IF .COB$$AL_WRITE_RAB [ .UNIT[0] ] EQL 0
985 2498 2          THEN
986 2499 2              !+
987 2500 2              Second parameter tells COB$$OPEN_IN whether VAX COBOL (0)
988 2501 2              or VAX RPG (1) is the caller.
989 2502 2              !-
990 2503 2              COB$$OPEN_IN ( .UNIT[0],
991 2504 2                  IF ( .FLAGS AND V_COB_RPG ) NEQ 0
992 2505 2                  THEN 1
993 2506 2                  ELSE 0 ) ;
994 2507 2
995 2508 2          RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
996 2509 2
997 2510 2          !+
998 2511 2          Find out if the device is a terminal.
999 2512 2          !-
1000 2513 2
1001 2514 2          BEGIN
1002 2515 2              LOCAL
1003 2516 2              STATUS,
1004 2517 2              NAM_DSC : REF BLOCK [8,BYTE] ;
1005 2518 2
1006 2519 2          NAM_DSC = .RAB + RAB$C_BLN ;
1007 2520 2
1008 2521 2          IF .COB$ACC_TERM_TYPE EQL 0
```



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COBSACC\_SCR - ACCEPT with screen enhancements

H 3  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 22  
(4)

```

1009 2522 3 THEN
1010 2523 4 IF NOT ( COB$$$SETUP_TERM_TYPE ( .NAM_DSC [DSC$A_POINTER],
1011 2524 4 .NAM_DSC [DSC$W_LENGTH],
1012 2525 4 COB$ACC_TERM_TYPE ) )
1013 2526 3 THEN LIB$STOP ( COB$_ERRDURACC ) ;
1014 2527 3
1015 2528 3 IF .COBSACC_TERM_TYPE EQL UNKNOWN
1016 2529 3 THEN
1017 2530 3 !+
1018 2531 3 | If terminal is UNKNOWN then it can be assumed we are working
1019 2532 3 | with files rather than terminals. Pull out of this routine
1020 2533 3 | and go to COB$$$ACC_SCR_FILE which uses a slightly different
1021 2534 3 | variation of the RMS $GET Service.
1022 2535 3 | -
1023 2536 4 BEGIN
1024 2537 4 STATUS = COB$$$ACC_SCR_FILE ( .UNIT, .STRING_DEST, .FLAGS, .DEFAULT,
1025 2538 4 .LENGTH, .ACC_SIZE, PUT_HERE, .YES_CONV,
1026 2539 4 .YES_PROTECT, .YES_SIGN ) ;
1027 2540 4
1028 2541 4 !+ Free local string PUT_HERE
1029 2542 4 | -
1030 2543 5 IF NOT ( STR$FREE1_DX ( PUT_HERE ) )
1031 2544 4 THEN LIB$STOP ( COB$_ERRDURACC ) ;
1032 2545 4
1033 2546 5 IF (NOT .STATUS)
1034 2547 4 THEN
1035 2548 4 RETURN 0
1036 2549 4 ELSE
1037 2550 4 RETURN 1 ;
1038 2551 3 END ;
1039 2552 3
1040 2553 3 END ;
1041 2554 2
1042 2555 2 !+
1043 2556 2 | Flag to COBSACCEPT that COBSACC_SCR has been called. COBSACCEPT will
1044 2557 2 | have to perform a Carriage Return.
1045 2558 2 | -
1046 2559 2
1047 2560 2 ACC_SCR = 1 ;
1048 2561 2
1049 2562 2 BEGIN ! Begin $GET
1050 2563 2
1051 2564 2 !+
1052 2565 2 | VAX COBOL Version 1 / Version 3 Interaction.
1053 2566 2 | Advancing philosophy : <LF> $GET <CR>
1054 2567 2 | <LF> based on previous call.
1055 2568 2 | <CR> based on current ACCEPT using FLAGS bit 10.
1056 2569 2 | If previous call requires advancing then perform a linefeed. DISPLAY (DISP)
1057 2570 2 | and ACCEPT (ACC_ADV) with advancing. POS = call to module COB$POS_ERASE
1058 2571 2 | remembers what previous call was, if advancing then POS, if no advancing
1059 2572 2 | then POS_DNA
1060 2573 2 | -
1061 2574 2
1062 2575 4 IF ( .COB$$$AB_PREV[0] EQL DISP
1063 2576 4 OR .COB$$$AB_PREV[0] EQL POS
1064 2577 4 OR .COB$$$AB_PREV[0] EQL ACC_ADV )
1065 2578 3 THEN
```



```
1066 2579 3      !+
1067 2580 3      !- Echo linefeed to terminal
1068 2581 3      !-
1069 2582 3      COB$$RMS_PUT_BYTE ( LINE_FD, .FLAGS ) ;
1070 2583 3
1071 2584 3      !+
1072 2585 3      !- Did user request any terminal attributes (bold, blink, underline, reverse) ?
1073 2586 3      !- If so, call COB$$SET_ATTRIBUTES_ONLY to get escape sequence to turn
1074 2587 3      !- attributes on and off.
1075 2588 3      !- PUT_FLAG - first four bits (0-3) of FLAGS parameter.
1076 2589 3      !-
1077 2590 3
1078 2591 3      PUT_FLAG = .FLAGS AND FLAG_MASK ;
1079 2592 3
1080 2593 3      IF .PUT_FLAG NEQ 0
1081 2594 3      THEN
1082 2595 4          IF NOT ( COB$$SET_ATTRIBUTES_ONLY ( .COBSACC_TERM_TYPE, .PUT_FLAG,
1083 2596 4              ON_BUF [0], ON_LEN,
1084 2597 4              OFF_BUF [0], OFF_LEN ) )
1085 2598 4              THEN LIB$STOP ( COB$_ERRDURACC ) ;
1086 2599 3
1087 2600 3      !+
1088 2601 3      !- If requested, add sequence to ON_BUF to ring terminal bell.
1089 2602 3      !-
1090 2603 3
1091 2604 3      IF ( .FLAGS AND V_BELL ) NEQ 0
1092 2605 3      THEN
1093 2606 4          BEGIN
1094 2607 4              ON_BUF [ .ON_LEN ] = BELL ;
1095 2608 4              ON_LEN = .ON_LEN + 1 ;
1096 2609 4          END ;
1097 2610 3
1098 2611 3      !+
1099 2612 3      !- Check parameters to see if the CONTROL KEY FORMAT 4 ACCEPT has been
1100 2613 3      !- requested. If so, pull out of this routine and call COB$$FORMAT_FOUR
1101 2614 3      !- which uses a different Terminator Mask and does not need all the
1102 2615 3      !- enhancements in COBSACC_SCR.
1103 2616 3      !-
1104 2617 3
1105 2618 3      IF NOT NULLPARAMETER (KEY)
1106 2619 3      THEN
1107 2620 4          BEGIN
1108 2621 4              LOCAL
1109 2622 4                  KEY_LEN ;
1110 2623 4
1111 2624 4              KEY_LEN = .KEY [DSC$W_LENGTH] ;
1112 2625 4              STR$DUPL_CHAR ( .KEY, KEY_LEN, BLANKS ) ;
1113 2626 4
1114 2627 4      !+
1115 2628 4      !- If these parameters are not present then we are dealing with
1116 2629 4      !- a Format Four ACCEPT rather than a Format Three ACCEPT.
1117 2630 4      !-
1118 2631 4
1119 2632 5      IF (NULLPARAMETER (LENGTH) AND
1120 2633 5          NULLPARAMETER (SIZE) AND
1121 2634 5          NULLPARAMETER (DEFAULT) AND
1122 2635 5          NULLPARAMETER (STRING_DEST) )
```



```
1123 2636 4      THEN
1124 2637 5      IF NOT ( COB$$$FORMAT_FOUR ( .UNIT, .FLAGS, .KEY ))
1125 2638 4      THEN RETURN 0
1126 2639 4      ELSE RETURN 1 ;
1127 2640 4      END ;
1128 2641 3
1129 2642 3
1130 2643 3  + Determine FUNC_VAL - QIO Function Modifiers used by RMS $GET Service.
1131 2644 3  Check FLAGS parameter to see if NO-ECHO was requested (bit 9), if so
1132 2645 3  set TRMSM_TM_NOECHO to suppress echoing of input characters to the terminal.
1133 2646 3  Set TRMSM_TM_ESCAPE to allow Escape Sequences to act as terminators (Arrow
1134 2647 3  keys, PF keys, and the Professional editing and top row function keys).
1135 2648 3  Set TRMSM_TM_NOFILTR to have the DELETE KEY handled by COB$$$DELETE_KEY.
1136 2649 3  Set TRMSM_TM_TRMNOECHO to suppress echoing of the termination character
1137 2650 3  (COB$$$AB_PREV handles advancing / no advancing).
1138 2651 3  -
1139 2652 3
1140 2653 3  IF ( .FLAGS AND V_NO_ECHO ) NEQ 0
1141 2654 3  THEN
1142 2655 4      BEGIN
1143 2656 4      FUNC_VAL = TRMSM_TM_ESCAPE + TRMSM_TM_NOFILTR + TRMSM_TM_TRMNOECHO
1144 2657 4      + TRMSM_TM_NOECHO ;
1145 2658 4      YES_NO_ECHO = 1 ;
1146 2659 4      END
1147 2660 3  ELSE
1148 2661 3      FUNC_VAL = TRMSM_TM_ESCAPE + TRMSM_TM_NOFILTR + TRMSM_TM_TRMNOECHO ;
1149 2662 3
1150 2663 3  +
1151 2664 3  Main Loop of routine.
1152 2665 3  PROT_OK = 1 -> there was no Protection error "plus"
1153 2666 3  CONV_OK = 1 -> there was no Conversion error "equal" SUCCESS -> pull out
1154 2667 3  of loop. Otherwise continue accepting data until there are no errors.
1155 2668 3  If error, reprompt user for more input via macro $ERROR_REPROMPT.
1156 2669 3  -
1157 2670 3
1158 2671 3  WHILE ( .PROT_OK EQL 0 ) OR ( .CONV_OK EQL 0 ) DO
1159 2672 4      BEGIN                                     ! Begin loop
1160 2673 4      LOCAL
1161 2674 4      TERM_SEEN : INITIAL (0) ;                 ! Flag for PROTECT check
1162 2675 4
1163 2676 4      IF .REPROMPT_DONE EQL 0
1164 2677 4      THEN
1165 2678 5      BEGIN                                     ! Begin no reprompt
1166 2679 5
1167 2680 5      +
1168 2681 5      If PROTECTION requested, put a Protected Field on the screen.
1169 2682 5      $PUT ACC_SIZE blanks to screen with attributes requested
1170 2683 5      by user turned on. (Escape sequences geared to VT100
1171 2684 5      terminals) Can only set a one line field as a max, therefore
1172 2685 5      FIELD_BUF holds up to 300 characters.
1173 2686 5      -
1174 2687 5
1175 2688 5      IF .COBSACC_TERM_TYPE EQL VT100
1176 2689 5      THEN
1177 2690 6      BEGIN                                     ! Begin VT100
1178 2691 6      IF .YES_PROTECT
1179 2692 6      THEN
```



```
: 1180      2693 7      BEGIN                      ! Begin Field
: 1181      2694 7      LOCAL
: 1182      2695 7      FIELD_BUF      : VECTOR [300, BYTE],
: 1183      2696 7      FIELD_LEN ;      ! size of FIELD_BUF
: 1184      2697 7
: 1185      2698 7      !+
: 1186      2699 7      Buffer FIELD_BUF to write Protected Field contains
: 1187      2700 7      - escape sequence to turn attributes on,
: 1188      2701 7      - number of blanks to write to screen and
: 1189      2702 7      - backspaces (same # as blanks) to put cursor
: 1190      2703 7      back to original position.
: 1191      2704 7      !-
: 1192      2705 7
: 1193      2706 7      CH$MOVE ( .ON_LEN, ON_BUF [0], FIELD_BUF [0] ) ;
: 1194      2707 7      FIELD_LEN = .ON_LEN ;
: 1195      2708 7      CH$FILL ( BLANK, .ACC_SIZE, FIELD_BUF [.FIELD_LEN] ) ;
: 1196      2709 7      FIELD_LEN = .FIELD_LEN + .ACC_SIZE ;
: 1197      2710 7      CH$FILL ( BS, .ACC_SIZE, FIELD_BUF [.FIELD_LEN] ) ;
: 1198      2711 7      FIELD_LEN = .FIELD_LEN + .ACC_SIZE ;
: 1199      2712 7
: 1200      2713 7      !+
: 1201      2714 7      If size of FIELD_BUF is greater than the size of the
: 1202      2715 7      maximum allowed for a $PUT buffer, issue an error
: 1203      2716 7      message. Issuing multiple $PUTs at this point does
: 1204      2717 7      not help as the cursor is unable to get back to the
: 1205      2718 7      starting position and ends up in the wrong line.
: 1206      2719 7      !-
: 1207      2720 7
: 1208      2721 8      IF .FIELD_LEN GTR (COB$K_ACC_SIZE - RMS_HEADER) ! 1024 -14
: 1209      2722 7      THEN
: 1210      2723 7      LIB$STOP ( COB$_ERRDURACC ) ;
: 1211      2724 7
: 1212      2725 7      !+
: 1213      2726 7      $PUT to write Protected Field to terminal
: 1214      2727 7      !-
: 1215      2728 7
: 1216      2729 7      COB$$RMS_PUT_BUFFER ( FIELD_BUF [0], .FIELD_LEN, .FLAGS ) ;
: 1217      2730 7
: 1218      2731 6      END ;                      ! End Field
: 1219      2732 5      END ;                      ! End VT100
: 1220      2733 5
: 1221      2734 5      !*****
: 1222      2735 5      !***** RMS $GET Service
: 1223      2736 5      !*****
: 1224      2737 5
: 1225      2738 5      !+
: 1226      2739 5      RMS $PUT to turn on terminal attributes (blink,bold,underline,reverse).
: 1227      2740 5      RMS $GET to accept input. Do not perform the $PUT if PROTECTED
: 1228      2741 5      is requested as the FIELD_BUF $PUT has already turned attributes
: 1229      2742 5      on.
: 1230      2743 5      Note : TRMS PROMPT not used because of buffer size limitations.
: 1231      2744 5      TRMS_MODIFIERS uses all of specified buffer for accepting input,
: 1232      2745 5      TRMS_PROMPT uses same buffer for both the prompt string and the
: 1233      2746 5      accepted data, therefore some space for accepting data is lost.
: 1234      2747 5      !-
: 1235      2748 5
: 1236      2749 5      IF .ON_LEN NEQ 0 AND .YES_PROTECT NEQ 1      ! If requested, turn
```



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$ACC\_SCR - ACCEPT with screen enhancements

L 3  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COB\$ACCEPT.B32;2

Page 26  
(4)

```
: 1237      2750 5      THEN                                     ! attributes on
: 1238      2751 5      COB$$RMS_PUT_BUFFER ( ON_BUF [0], .ON_LEN, .FLAGS ) ;
: 1239      2752 5
: 1240      2753 5      !+
: 1241      2754 5      !- RMS $GET to accept input from terminal.
: 1242      2755 5      !-
: 1243      2756 5
: 1244      2757 5      RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
: 1245      2758 5      COB$$RMS_GET ( .RAB, .FUNC_VAL, .ACC_SIZE,
: 1246      2759 5      .PUT_HERE [DSC$A_POINTER] ) ;
: 1247      2760 5
: 1248      2761 5      END                                     ! End of no reprompt
: 1249      2762 4      ELSE
: 1250      2763 4      REPROMPT_DONE = 0 ;                     ! re-set flag
: 1251      2764 4
: 1252      2765 4      !+
: 1253      2766 4      !- Get number of characters read and terminator size from the fields
: 1254      2767 4      !- of the RAB. Pass this info along to other routines.
: 1255      2768 4      !- RAB fields -
: 1256      2769 4      !-     rab [rab$l_sts]           = status
: 1257      2770 4      !-     rab [rab$l_rsz]           = x         no. of chars read
: 1258      2771 4      !-     rab [cob$$b_stv0_term] = d         <cr> terminator seen
: 1259      2772 4      !-     rab [cob$$b_stv2_len] = 1         size of terminator
: 1260      2773 4      !- Save this information before COB$$PARTIAL_SEQ does any more $GETs.
: 1261      2774 4      !-
: 1262      2775 4
: 1263      2776 4      CHARS_READ = .RAB [RAB$W_RSZ] ;         ! Number of chars read
: 1264      2777 4      TERM_SIZE = .RAB [COB$$B_STV2_LEN] ;    ! Size and location of
: 1265      2778 4      TERM_LOC  = .RAB [COB$$B_STV0_TERM] ;    ! terminator - other
: 1266      2779 4      !-                                     ! routines may update
: 1267      2780 4      !-                                     ! these
: 1268      2781 4
: 1269      2782 4      !+
: 1270      2783 4      !- Check for partial sequence error - not enough room in input buffer
: 1271      2784 4      !- to hold entire escape sequence when a Protected ACCEPT is performed.
: 1272      2785 4      !- If necessary, call COB$$PARTIAL_SEQ to read remainder of sequence.
: 1273      2786 4      !-
: 1274      2787 4      IF .RAB [RAB$L_STS] EQL RMSS_PES
: 1275      2788 4      THEN
: 1276      2789 5      BEGIN
: 1277      2790 5      TERM_SEEN = 1 ;                         ! Set flag here as
: 1278      2791 5      COB$$PARTIAL_SEQ ( PARAMETERS, .UNIT ) ; ! COB$$PARTIAL_SEQ may
: 1279      2792 4      END ;                                   ! change status value
: 1280      2793 4
: 1281      2794 4      !+
: 1282      2795 4      !- If terminator was the DELETE KEY call COB$$DELETE_KEY.
: 1283      2796 4      !-
: 1284      2797 4
: 1285      2798 4      IF .RAB [COB$$B_STV0_TERM] EQL DEL_KEY
: 1286      2799 4      THEN
: 1287      2800 4      COB$$DELETE_KEY ( PARAMETERS, .UNIT, .FLAGS ) ;
: 1288      2801 4
: 1289      2802 4      !*****
: 1290      2803 4      !***** PROTECTED
: 1291      2804 4      !*****
: 1292      2805 4
: 1293      2806 4      !+
```



```
1294 2807 4      | Was terminator seen on PROTECTED READ?
1295 2808 4
1296 2809 4      | Looking for terminator to make sure that user hasn't tried to go
1297 2810 4      | beyond the bounds of a PROTECTED READ.
1298 2811 4
1299 2812 4      | Two ways for a protected read to complete -
1300 2813 4      | 1. terminator typed before buffer filled ( no further check necessary)
1301 2814 4      | 2. buffer fill ( no terminator seen)
1302 2815 4      |     - do a one character read to make sure terminator is
1303 2816 4      |       typed, not another character.
1304 2817 4
1305 2818 4      | The following RAB fields look like this if buffer filled
1306 2819 4      |       rab [rab$l_sts]      = status
1307 2820 4      |       rab [rab$l_rsz]     = x      no. of chars read (acc_size)
1308 2821 4      |       rab [cob$$b_stv0_term] = 0    no terminator seen
1309 2822 4      |       rab [cob$$b_stv2_len] = 0    size of terminator
1310 2823 4
1311 2824 4
1312 2825 5      | IF (.YES_PROTECT )           | Was PROTECTION requested?
1313 2826 5      |       AND ( .CHARS_READ NEQ 0 ) | AND is it needed
1314 2827 4      | THEN
1315 2828 4      | IF .RAB [RAB$l_STS] EQL RMSS$TNS | RMSS$TNS = Terminator
1316 2829 4      |       AND .TERM_SEEN EQL 0      | Not Seen
1317 2830 4      | THEN
1318 2831 5      |       BEGIN                     | Begin protect check $GET
1319 2832 5      |       +
1320 2833 5      |       | After initial $GET is performed it is necessary to perform a
1321 2834 5      |       | $GET of length 1 to make sure that there are no characters
1322 2835 5      |       | typed by the user that exceed the maximum allowed.
1323 2836 5      |       | (Do not echo character to terminal.)
1324 2837 5      |       | If the $GET of one character results in a terminator, there
1325 2838 5      |       | is no problem.
1326 2839 5      |       | If the $GET of one character results in an attempt to type
1327 2840 5      |       | extra characters, there is an error.
1328 2841 5      |       |
1329 2842 5      |       | If VAX RPG is the caller, always return control to the
1330 2843 5      |       | calling program on an error.
1331 2844 5      |       -
1332 2845 5
1333 2846 5      | LOCAL
1334 2847 5      |       NO_CHAR : INITIAL (0),      | =1 no Protection error
1335 2848 5      |       HAVE_TERM : INITIAL (0);    | =1 terminator seen
1336 2849 5
1337 2850 5      | WHILE .HAVE_TERM NEQ 1 DO
1338 2851 6      |       BEGIN                     | Begin HAVE_TERM loop
1339 2852 6
1340 2853 6      |       NO_CHAR = 0 ;
1341 2854 6      |       FUNC_VAL_2 = TRMSM_TM_ESCAPE + TRMSM_TM_NOFILTR
1342 2855 6      |                   + TRMSM_TM_TRMNOECHO + TRMSM_TM_NOECHO ;
1343 2856 6
1344 2857 6      |       RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
1345 2858 6      |       COB$$RMS_GET (".RAB",.FUNC_VAL_2, 1, NEXT_CHAR ) ;
1346 2859 6
1347 2860 6      |       +
1348 2861 6      |       | If user did not attempt to enter more data, set TERM_SIZE
1349 2862 6      |       | and TERM_IN_NEXT before possible call to COB$$PARTIAL_SEQ.
1350 2863 6      |       | If not enough room in $GET buffer to hold entire escape
```



```
: 1351      2864 6      | sequence then call COB$$PARTIAL_SEQ to read remainder
: 1352      2865 6      | of sequence.
: 1353      2866 6      |
: 1354      2867 6      |
: 1355      2868 6      | IF .RAB [RAB$W_RSZ] EQL 0      ! No more data entered.
: 1356      2869 6      | THEN
: 1357      2870 7      |     BEGIN
: 1358      2871 7      |         NO_CHAR = 1 ;          ! Move terminator into
: 1359      2872 7      |         NEXT_CHAR [0] = .RAB [COB$$B_STV0_TERM] ; ! NEXT_CHAR
: 1360      2873 6      |     END
: 1361      2874 6      |     TERM_SIZE = .RAB [COB$$B_STV2_LEN] ; ! Terminator size.
: 1362      2875 6      |     TERM_LOC = .RAB [COB$$B_STV0_TERM] ; ! Terminator location.
: 1363      2876 6      |     TERM_IN_NEXT = 1 ;        ! Terminator on NEXT_CHAR
: 1364      2877 6      | IF .RAB [RAB$L_STS] EQL RMSS_PES
: 1365      2878 6      | THEN
: 1366      2879 6      |     COB$$PARTIAL_SEQ ( PARAMETERS, .UNIT ) ;
: 1367      2880 6      |
: 1368      2881 6      | + Terminators are the only acceptable input at this point.
: 1369      2882 6      | If NO_CHAR = 1 then there is no Protection error.
: 1370      2883 6      |
: 1371      2884 6      |
: 1372      2885 6      |
: 1373      2886 6      | IF .NO_CHAR
: 1374      2887 6      | THEN
: 1375      2888 7      |     BEGIN                    ! Begin TERM accepted
: 1376      2889 7      |     PROT_OK = 1 ;           ! $GET successful
: 1377      2890 7      |     HAVE_TERM = 1 ;
: 1378      2891 7      |
: 1379      2892 7      | *****
: 1380      2893 7      | ***** DELETE KEY
: 1381      2894 7      | *****
: 1382      2895 7      |
: 1383      2896 7      | + Was terminator the DELETE KEY ? If so, call
: 1384      2897 7      | COB$$DELETE_KEY to erase the last character
: 1385      2898 7      | read and to continue reading for input.
: 1386      2899 7      |
: 1387      2900 7      |
: 1388      2901 7      | IF .RAB [COB$$B_STV0_TERM] EQL DEL_KEY
: 1389      2902 7      | THEN
: 1390      2903 8      |     BEGIN
: 1391      2904 8      |     COB$$DELETE_KEY ( PARAMETERS, .UNIT, .FLAGS ) ;
: 1392      2905 8      |     +
: 1393      2906 8      |     | Check to see if we fell out of COB$$DELETE_KEY
: 1394      2907 8      |     | without a valid terminator. If so, keep
: 1395      2908 8      |     | looking for it.
: 1396      2909 8      |     |
: 1397      2910 8      |     IF .TERM_SIZE EQL 0
: 1398      2911 8      |     THEN
: 1399      2912 9      |         BEGIN
: 1400      2913 9      |         HAVE_TERM = 0 ;      ! Loop again
: 1401      2914 9      |         PROT_OK = 0 ;
: 1402      2915 9      |         END
: 1403      2916 8      |     ELSE
: 1404      2917 9      |         BEGIN
: 1405      2918 9      |         HAVE_TERM = 1 ;
: 1406      2919 9      |         TERM_IN_NEXT = 0 ;   ! Note - COB$$DELETE_KEY put
: 1407      2920 8      |         END ;               ! the terminator in
```



```
: 1408      2921  8                                ! PUT_HERE.
: 1409      2922  7                                END ;
: 1410      2923  7
: 1411      2924  7                                ! End TERM accepted
: 1412      2925  6                                ELSE
: 1413      2926  6
: 1414      2927  6 !*****
: 1415      2928  6 !***** PROTECTION ERROR
: 1416      2929  6 !*****
: 1417      2930  6
: 1418      2931  6
: 1419      2932  6
: 1420      2933  6
: 1421      2934  6
: 1422      2935  6
: 1423      2936  6
: 1424      2937  6
: 1425      2938  7
: 1426      2939  7
: 1427      2940  7
: 1428      2941  7
: 1429      2942  7
: 1430      2943  7
: 1431      2944  5
: 1432      2945  5
: 1433      2946  4
: 1434      2947  4
: 1435      2948  4
: 1436      2949  4
: 1437      2950  4
: 1438      2951  4
: 1439      2952  4
: 1440      2953  4
: 1441      2954  4
: 1442      2955  4
: 1443      2956  4
: 1444      2957  4
: 1445      2958  4 !*****
: 1446      2959  4 !***** CONTROL KEY
: 1447      2960  4 !*****
: 1448      2961  4
: 1449      2962  4
: 1450      2963  4
: 1451      2964  4
: 1452      2965  4
: 1453      2966  4
: 1454      2967  5
: 1455      2968  5
: 1456      2969  5
: 1457      2970  5
: 1458      2971  5
: 1459      2972  5
: 1460      2973  5
: 1461      2974  5
: 1462      2975  5
: 1463      2976  5
: 1464      2977  5

                                ! End HAVE_TERM loop
                                ! End protect check $GET

                                !+
                                ! Protection requested but terminator already seen,
                                ! no need for 1 character Read.
                                !-
                                PROT_OK = 1

                                !+
                                ! Protection not requested, no need for 1 character Read.
                                !-
                                PROT_OK = 1 ;

                                !*****
                                !***** CONTROL KEY
                                !*****

                                IF .PROT_OK
                                THEN
                                    !+
                                    ! No sense going thru Control Key code if there was a protection error.
                                    !-
                                    BEGIN
                                        ! Begin Control Key

                                        IF .TERM_IN_NEXT
                                        THEN
                                            ! Locate terminator,
                                            ! which buffer is it in.
                                            TERM_PTR = NEXT_CHAR[0]
                                        ELSE
                                            TERM_PTR = .PUT_HERE[DSC$A_POINTER] + .CHARS_READ ;

                                    !+
                                    ! If parameter KEY not sent (.KEY = 0) then CR, TAB, CONTROL Z,
                                    ! and DELETE KEY are the only legal terminators.
```



```
1465 2978 5
1466 2979 5
1467 2980 5
1468 2981 5
1469 2982 5
1470 2983 5
1471 2984 5
1472 2985 5
1473 2986 5
1474 2987 5
1475 2988 5
1476 2989 5
1477 2990 5
1478 2991 5
1479 2992 6
1480 2993 6
1481 2994 6
1482 2995 6
1483 2996 6
1484 2997 6
1485 2998 6
1486 2999 6
1487 3000 6
1488 3001 6
1489 3002 6
1490 3003 6
1491 3004 6
1492 3005 6
1493 3006 6
1494 3007 6
1495 3008 6
1496 3009 6
1497 3010 6
1498 3011 6
1499 3012 7
1500 3013 7
1501 3014 7
1502 3015 8
1503 3016 8
1504 3017 8
1505 3018 8
1506 3019 8
1507 3020 8
1508 3021 8
1509 3022 8
1510 3023 7
1511 3024 8
1512 3025 8
1513 3026 8
1514 3027 8
1515 3028 8
1516 3029 8
1517 3030 8
1518 3031 7
1519 3032 6
1520 3033 6
1521 3034 6

      If parameter KEY not 0 then CR, TAB, CONTROL Z, DELETE KEY, PF,
      ARROW and SPECIAL FUNCTION PROFESSIONAL Keys are legal
      terminators. Copy terminator to KEY parameter.

      Special treatment needed for CONTROL Z under RMS. There is a
      difference between ^Z being typed alone and with data.
      When ^Z is typed with data the ^Z is stored in
      RAB[RAB$STVO_TERM], but when ^Z is typed alone the status
      RMSS_EOF is returned from the $Get Service.

      IF .TERM_SIZE EQL 1                                ! One byte terminator
      THEN
      BEGIN
      TERM_PTR = RAB [COB$$B_STVO_TERM] ;
      SELECTONE .RAB [COB$$B_STVO_TERM] OF
      SET
      [ CR,                                     ! Carriage Return
      TAB ] :                                       ! Tab
      +
      These keys are legal, do nothing if KEY = 0.
      -
      IF NOT NULLPARAMETER (KEY)
      THEN
      CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;

      [ CZ ] :                                       ! Control z
      +
      CONTROL Z hit along with data ( terminator in
      RAB [COB$$B_STVO_TERM] )
      -
      BEGIN
      IF (.FLAGS AND V_COB_RPG) NEQ 0
      THEN
      BEGIN
      +
      VAX RPG - Control Z is an illegal terminator.
      -
      LEGAL = 0 ;
      COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS,
      .KEY ) ;
      END
      ELSE
      BEGIN
      +
      VAX COBOL - Control Z has special meaning.
      -
      COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
      COB$$CONTROL_Z ( .UNIT, .KEY ) ;
      RETURN 0 ;
      END ;
      END ;

      [ DEL_KEY ] :                                       ! Delete key
```



```
1522 3035 6
1523 3036 7
1524 3037 7
1525 3038 6
1526 3039 6
1527 3040 6
1528 3041 6
1529 3042 7
1530 3043 7
1531 3044 7
1532 3045 7
1533 3046 6
1534 3047 6
1535 3048 6
1536 3049 6
1537 3050 5
1538 3051 5
1539 3052 5
1540 3053 6
1541 3054 6
1542 3055 6
1543 3056 6
1544 3057 6
1545 3058 6
1546 3059 6
1547 3060 7
1548 3061 7
1549 3062 7
1550 3063 7
1551 3064 6
1552 3065 7
1553 3066 7
1554 3067 7
1555 3068 7
1556 3069 6
1557 3070 6
1558 3071 5
1559 3072 6
1560 3073 6
1561 3074 6
1562 3075 6
1563 3076 6
1564 3077 6
1565 3078 6
1566 3079 6
1567 3080 6
1568 3081 6
1569 3082 6
1570 3083 6
1571 3084 6
1572 3085 6
1573 3086 7
1574 3087 8
1575 3088 7
1576 3089 8
1577 3090 8
1578 3091 8

      BEGIN
      COB$$DELETE_KEY ( PARAMETERS, .UNIT, .FLAGS ) ;
      END ;

      [OTHERWISE] :
      ! Error - key not a
      ! terminator
      BEGIN
      LEGAL = 0 ;
      COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS,
      .KEY ) ;
      END ;
      TES ;
      END
ELSE
  IF .CHARS_READ EQL 0 AND .RAB [RAB$$_STS] EQL RMS$_EOF
  THEN
    BEGIN
    !+
    ! CONTROL Z hit alone - terminator not placed in
    ! RAB [COB$$B_STVO_TERM], but signaled via RAB [RAB$$_STS].
    !-
    IF (.FLAGS AND V_COB_RPG) NEQ 0
    THEN
      BEGIN
      LEGAL = 0 ;
      COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS, .KEY ) ;
      END
    ELSE
      BEGIN
      ! VAX COBOL
      COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
      COB$$CONTROL_Z ( .UNIT, .KEY ) ;
      RETURN 0 ;
      END
    END
  ELSE
    BEGIN
    !+
    ! Escape Sequence as Terminator. .TERM_SIZE greater
    ! than 1 and RMS$_EOF not signaled.
    !-
    IF NOT NULLPARAMETER (KEY)
    THEN
      !+
      ! COB$$CONTROL_KEY converts terminator sequences to
      ! COBOL defined sequences and fills in KEY parameter
      ! if terminator is legal.
      !-
      BEGIN
      IF NOT ( COB$$CONTROL_KEY (TERM_PTR, .TERM_SIZE, .KEY) )
      THEN
        BEGIN
        LEGAL = 0 ;
        COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS,
```



```
1579 3092 8
1580 3093 7
1581 3094 7
1582 3095 6
1583 3096 6
1584 3097 6
1585 3098 6
1586 3099 6
1587 3100 7
1588 3101 7
1589 3102 7
1590 3103 6
1591 3104 5
1592 3105 4
1593 3106 4
1594 3107 4
1595 3108 4
1596 3109 4
1597 3110 4
1598 3111 4
1599 3112 4
1600 3113 4
1601 3114 4
1602 3115 4
1603 3116 4
1604 3117 4
1605 3118 4
1606 3119 4
1607 3120 4
1608 3121 4
1609 3122 4
1610 3123 5
1611 3124 4
1612 3125 4
1613 3126 4
1614 3127 4
1615 3128 4
1616 3129 5
1617 3130 5
1618 3131 5
1619 3132 4
1620 3133 4
1621 3134 5
1622 3135 4
1623 3136 5
1624 3137 4
1625 3138 5
1626 3139 5
1627 3140 5
1628 3141 5
1629 3142 5
1630 3143 5
1631 3144 5
1632 3145 5
1633 3146 5
1634 3147 5
1635 3148 5

      END ;
    ELSE
      + Terminator of size greater than 1 is illegal when KEY is
      - not used.
      BEGIN
        LEGAL = 0 ;
        COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS, .KEY ) ;
      END ;
    END ;

      END ;                                ! End Control Key

      *****
      ***** NULL INPUT
      *****

      +
      - Null input
      RAB fields look like this for null input
          rab [rab$l_sts]      = 1      status
          rab [rab$l_rsz]     = 0      no. of chars read
          rab [cob$$b_stv0_term] = d    <cr> terminator seen
          rab [cob$$b_stv2_len] = 1    size of terminator

      - Check for DEFAULT parameter - if present prepare to put it through
      Conversion routines by placing DEFAULT in PUT_HERE.

      IF ( .CHARS_READ EQL 0 ) AND ( ( .FLAGS AND V_COB_RPG ) NEQ 0 )
      THEN
        +
        - In case of null input for RPG, simply return (no DEFAULT).
        But perform any necessary clean up first.
        BEGIN
          COB$$RPG_CLEAN_UP ( .FLAGS ) ;
          RETURN 1 ;
        END ;

      IF ( .CHARS_READ EQL 0 )
      THEN
        IF NOT NULLPARAMETER ( DEFAULT ) AND ( .YES_DEFAULT EQL 0 )
        THEN
          BEGIN                                ! Begin DEFAULT
            CHARS_READ = .DEFAULT [DSC$W_LENGTH];
            YES_DEFAULT = 1 ;

            +
            - Protection check for DEFAULT excluding the Floating
            Point data types ( these will be handled in routine
            COB$$VERIFY_FL_RANGE ).
          END ;
```



```
1636 3149 6      IF (.YES_PROTECT AND
1637 3150 7          ( .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_F AND
1638 3151 6          .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_D ))
1639 3152 5      THEN
1640 3153 5          +
1641 3154 5          If the length of DEFAULT is greater than the expected
1642 3155 5          input size ACC_SIZE, then there is a Protection error.
1643 3156 5          This should be caught at compile time by VAX COBOL and
1644 3157 5          a fatal error message issued, however there is one case
1645 3158 5          that the compiler cannot catch, therefore issue a fatal
1646 3159 5          run time error here.
1647 3160 5          -
1648 3161 6          IF (.DEFAULT [DSC$W_LENGTH] GTR .ACC_SIZE)
1649 3162 5          THEN
1650 3163 5              LIB$STOP ( COB$_INVDEFVAL )
1651 3164 5          ELSE
1652 3165 5              PROT_OK = 1                      ! No PROTECT error
1653 3166 5          ELSE
1654 3167 5              PROT_OK = 1 ;                      ! No PROTECT error
1655 3168 5          END ;                      ! End DEFAULT
1656 3169 4
1657 3170 4
1658 3171 4  !*****
1659 3172 4  !***** CONVERSION
1660 3173 4  !***** ALL RMS $GETs COMPLETED EXCEPT POSSIBLE REPROMPT ON A CONVERSION ERROR.
1661 3174 4  !*****
1662 3175 4
1663 3176 4  +
1664 3177 4  If conversion requested, call routine COB$$$ACC_CONVERT
1665 3178 4  -
1666 3179 4
1667 3180 5      IF ( .PROT_OK )                      ! If protection error,
1668 3181 4      THEN                          ! don't go thru conversion
1669 3182 5          IF ( .YES_CONV )
1670 3183 4          THEN
1671 3184 4              CONV_OK = COB$$$ACC_CONVERT ( .STRING_DEST, .FLAGS,
1672 3185 4              .DEFAULT, PUT_HERE, .CHARS_READ,
1673 3186 4              .YES_DEFAULT, .YES_SIGN )
1674 3187 4          ELSE
1675 3188 5              BEGIN
1676 3189 5                  LOCAL
1677 3190 5                  COPY_NUM ;
1678 3191 5                  +
1679 3192 5                  No conversion requested - copy input data to STRING_DEST.
1680 3193 5                  Use STR$COPY_R because it BLANK fills.
1681 3194 5                  -
1682 3195 5
1683 3196 5                  IF .CHARS_READ LSS .STRING_DEST[DSC$W_LENGTH]
1684 3197 5                  THEN
1685 3198 5                      COPY_NUM = .CHARS_READ
1686 3199 5                  ELSE
1687 3200 5                      COPY_NUM = .STRING_DEST[DSC$W_LENGTH] ;
1688 3201 5
1689 3202 5                  STR$COPY_R ( .STRING_DEST, COPY_NUM,
1690 3203 6                  (IF .YES_DEFAULT
1691 3204 6                  THEN .DEFAULT [DSC$A_POINTER]
1692 3205 5                  ELSE .PUT_HERE [DSC$A_POINTER] )) ;
```



```

: 1693      3206 5
: 1694      3207 5
: 1695      3208 5
: 1696      3209 4
: 1697      3210 4
: 1698      3211 4
: 1699      3212 4
: 1700      3213 4
: 1701      3214 4
: 1702      3215 4
: 1703      3216 4
: 1704      3217 4
: 1705      3218 4
: 1706      3219 4
: 1707      3220 4
: 1708      3221 4
: 1709      3222 4
: 1710      3223 4
: 1711      3224 4
: 1712      3225 4
: 1713      3226 4
: 1714      3227 4
: 1715      3228 5
: 1716      3229 5
: 1717      3230 5
: 1718      3231 5
: 1719      3232 5
: 1720      3233 5
: 1721      3234 5
: 1722      3235 6
: 1723      3236 6
: 1724      3237 6
: 1725      3238 6
: 1726      3239 5
: 1727      3240 5
: 1728      3241 5
: 1729      3242 5
: 1730      3243 6
: 1731      3244 6
: 1732      3245 6
: 1733      3246 6
: 1734      3247 6
: 1735      3248 6
: 1736      3249 6
: 1737      3250 5
: 1738      3251 5
: 1739      3252 5
: 1740      3253 5
: 1741      3254 5
: 1742      3255 5
: 1743      3256 5
: 1744      3257 6
: 1745      3258 7
: 1746      3259 6
: 1747      3260 7
: 1748      3261 7
: 1749      3262 7

      CONV_OK = 1 ;                      ! set CONV_OK to success
    END;

    + Conversion completed - was it successful ?
    -
  IF .CONV_OK EQL 0
  THEN
    + CONVERSION error. Read UNIT parameter to determine what
    to do.
      Byte 2 of                               Conversion
      UNIT                                           error
      0                                           reprompt
      1 ( at end )                             reprompt
      2 ( on exception )                       return
    -
  BEGIN                                           ! Begin conversion error
  IF ( .FLAGS AND V_COB_RPG ) NEQ 0
  THEN
    + VAX RPG - return on a Conversion Error, ring bell
    and clean up before exiting.
    -
    BEGIN
    COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
    COB$$RPG_CLEAN_UP ( .FLAGS ) ;
    RETURN 0 ;
    END ;
  IF .UNIT [1] EQL 2
  THEN
    BEGIN
    + Clean up before returning control to VAX COBOL.
    -
    COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
    RETURN 0 ;
    END
  ELSE
    + Reprompt
    - sound terminal bell,
    - clear screen of all typed characters,
    - reset cursor to original line/column position
    -
    BEGIN
    IF ((.FLAGS AND V_NO_ECHO) NEQ 0 ) OR ( .YES_DEFAULT )
    THEN
      BEGIN
      COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ; ! No re-positioning
      ! necessary
      END
    END
  END
```



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COBSACC\_SCR - ACCEPT with screen enhancements

H 4  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 35  
(4)

```
: 1750      3263 6      ELSE
: 1751      3264 7      BEGIN
: 1752      3265 7      $ERROR_REPROMPT ;      ! Re-position
: 1753      3266 6      END ;
: 1754      3267 6      CONV_OK = 0 ;      ! Signal Conversion error
: 1755      3268 6      YES_DEFAULT = 0 ;
: 1756      3269 5      END ;
: 1757      3270 5      END      ! End conversion error
: 1758      3271 5
: 1759      3272 4      ELSE
: 1760      3273 4      +
: 1761      3274 4      Conversion not done either because Protection failed (PROT_OK=0)
: 1762      3275 4      or there was no data to convert
: 1763      3276 4      -
: 1764      3277 4      CONV_OK = 1 ;
: 1765      3278 4
: 1766      3279 3      END ;      ! End loop
: 1767      3280 3
: 1768      3281 3      +
: 1769      3282 3      RMS $GET complete - fill in optional LENGTH parameter with the
: 1770      3283 3      number of characters read.
: 1771      3284 3      -
: 1772      3285 3
: 1773      3286 3      IF NOT NULLPARAMETER (LENGTH)
: 1774      3287 3      THEN
: 1775      3288 3      .LENGTH = .CHARS_READ ;
: 1776      3289 3
: 1777      3290 3      +
: 1778      3291 3      ***** CLEAN UP
: 1779      3292 3      *****
: 1780      3293 3
: 1781      3294 3      +
: 1782      3295 3      Call COB$$CLEAN_UP to perform (if needed) cursor positioning,
: 1783      3296 3      turn off terminal attributes, and advancing.
: 1784      3297 3      -
: 1785      3298 3
: 1786      3299 3      COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
: 1787      3300 3
: 1788      3301 3      END;      ! End $GET
: 1789      3302 3
: 1790      3303 3      +
: 1791      3304 3      Free local strings PUT_HERE
: 1792      3305 3      -
: 1793      3306 3
: 1794      3307 3      IF NOT ( STR$FREE1 DX ( PUT_HERE ) )
: 1795      3308 3      THEN LIB$STOP ( COB$_ERRDURACC ) ;
: 1796      3309 3
: 1797      3310 3
: 1798      3311 3      RETURN 1;
: 1799      3312 1      END;      ! end of routine COB$ACC_SCR
```

00000000# 001DE 001E0 P.AAR: .BLKB 2  
.LONG 0[22]



				OFFC 00000	.ENTRY	COBSACC_SCR, Save R2,R3,R4,R5,R6,R7,R8,R9,-	
		5E	EBF0	CE 9E 00002	MOVAB	R10,R11	2181
			OC	AE D4 00007	CLRL	-5136(SP), SP	
				5A D4 0000A	CLRL	ON LEN	2271
				7E 7C 0000C	CLRL	PROT OK	
				7E 7C 0000E	CLRL	REPRMPT DONE	
				01 DD 00010	CLRL	YES_NO_ECHO	
				56 D4 00012	PUSHL	#1	
94	AD	8E	AF 0058	8F 28 00014	CLRL	P DATA_TYPE	
				50 D4 0001C	MOVBC3	#88, P.AAR, PARAMETERS	2297
		24	AE	20 D0 0001E	CLRL	ZEROES	2312
		94	AD 02100000	8F D0 00022	MOVL	#32, BLANKS	2313
			98	AD D4 0002A	MOVL	#34603008, PUT_HERE	2319
			OC	AC D0 0002D	CLRL	PUT_HERE+4	2322
		59		05 E1 00031	MOVL	FLAGS, R9	2337
04		59		01 D0 00035	BBC	#5, R9, 1\$	
		08	AE	59 95 00039 1\$:	MOVL	#1, YES_CONV	
				02 18 0003B	TSTB	R9	2338
				6E D4 0003D	BGEQ	2\$	
03		59		08 E0 0003F 2\$:	CLRL	YES_SIGN	
			00FC	31 00043	BBS	#8, R9, 3\$	2340
		DO	AD	01 D0 00046 3\$:	BRW	20\$	
			14	AC D5 0004A	MOVL	#1, YES_PROTECT	2343
				08 13 0004D	TSTL	SIZE	2344
		AC	AD	AC B0 0004F	BEQL	5\$	
			00F1	31 00054 4\$:	MOVW	SIZE, ACC_SIZE	2345
				52 D4 00057 5\$:	BRW	21\$	
		50	08	AC D0 00059	CLRL	PP99	2347
		52	09	AO 9A 0005D	MOVL	STRING_DEST, R0	2352
		51	08	AO 98 00061	MOVZBL	9(R0), PP99	
		52		51 C0 00065	CVTBL	8(R0), R1	
		AC	AD	60 B0 00068	ADDL2	R1, PP99	
				54 D4 0006C	MOVW	(R0), ACC_SIZE	2353
		09	03	AO 91 0006E	CLRL	R4	2361
				6B 12 00072	CMPB	3(R0), #9	
				54 D6 00074	BNEQ	13\$	
				52 D5 00076	INCL	R4	
				05 19 00078	TSTL	PP99	2362
				AO 95 0007A	BLSS	6\$	
			08	60 15 0007D	TSTB	8(R0)	2363
		56		01 D0 0007F 6\$:	BLEQ	13\$	
				53 D4 00082	MOVL	#1, P_DATA_TYPE	2366
				52 D5 00084	CLRL	R3	2367
				11 18 00086	TSTL	PP99	
				53 D6 00088	BGEQ	8\$	
		51	08	AO 98 0008A	INCL	R3	
				03 18 0008E	CVTBL	8(R0), R1	2369
				51 CE 00090	BGEQ	7\$	
		AC	AD	51 B0 00093 7\$:	MNEGL	R1, R1	
				04 11 00097	MOVW	R1, ACC_SIZE	
		AC	AD	52 B0 00099 8\$:	BRB	9\$	2371
		B3	08	AE E9 0009D 9\$:	MOVW	PP99, ACC_SIZE	2373
		03		53 E9 000A1	BLBC	YES_CONV, -4\$	2379
			AC	AD B6 000A4	BLBC	R3, -10\$	2381
		51	02	AO 9A 000A7 10\$:	INCW	ACC_SIZE	2389
					MOVZBL	2(R0), R1	



OF		51	91	000AB		CMPB	R1	#15		
		03	13	000AE		BEQL	11\$			
	AC	AD	B6	000B0		INCW	ACC_SIZE			2391
07		51	91	000B3	11\$:	CMPB	R1	#7		2397
		1E	13	000B6		BEQL	12\$			
03		51	91	000B8		CMPB	R1	#3		2398
		19	13	000BB		BEQL	12\$			
08		51	91	000BD		CMPB	R1	#8		2399
		14	13	000C0		BEQL	12\$			
04		51	91	000C2		CMPB	R1	#4		2400
		0F	13	000C5		BEQL	12\$			
09		51	91	000C7		CMPB	R1	#9		2401
		0A	13	000CA		BEQL	12\$			
05		51	91	000CC		CMPB	R1	#5		2402
		05	13	000CF		BEQL	12\$			
15		51	91	000D1		CMPB	R1	#21		2403
		72	12	000D4		BNEQ	21\$			
		6E	D5	000D6	12\$:	TSTL	YES_SIGN			2404
		6E	12	000D8		BNEQ	21\$			
	AC	AD	B7	000DA		DECW	ACC_SIZE			2406
		69	11	000DD		BRB	21\$			2361
65	08	AE	E9	000DF	13\$:	BLBC	YES_CONV, 21\$			2415
51	02	A0	9A	000E3		MOVZBL	2(R0), R1			2423
13		51	91	000E7		CMPB	R1	#19		
		0D	13	000EA		BEQL	14\$			
11		51	91	000EC		CMPB	R1	#17		2424
		08	13	000EF		BEQL	14\$			
15		51	91	000F1		CMPB	R1	#21		2425
		06	12	000F4		BNEQ	15\$			
03		6E	E9	000F6		BLBC	YES_SIGN, 15\$			2426
	AC	AD	B6	000F9	14\$:	INCW	ACC_SIZE			2428
29		54	E9	000FC	15\$:	BLBC	R4, 17\$			2438
07		51	91	000FF		CMPB	R1	#7		2440
		19	13	00102		BEQL	16\$			
03		51	91	00104		CMPB	R1	#3		2441
		14	13	00107		BEQL	16\$			
08		51	91	00109		CMPB	R1	#8		2442
		0F	13	0010C		BEQL	16\$			
04		51	91	0010E		CMPB	R1	#4		2443
		0A	13	00111		BEQL	16\$			
09		51	91	00113		CMPB	R1	#9		2444
		05	13	00116		BEQL	16\$			
05		51	91	00118		CMPB	R1	#5		2445
		0B	12	0011B		BNEQ	17\$			
08		6E	E9	0011D	16\$:	BLBC	YES_SIGN, 17\$			2446
AC	AD	09	A0	9B	00120	MOVZBW	9(R0), ACC_SIZE			2448
	AC	AD	B6	00125		INCW	ACC_SIZE			
0A		51	91	00128	17\$:	CMPB	R1	#10		2454
		04	12	0012B		BNEQ	18\$			
AC	AD	0D	B0	0012D		MOVW	#13, ACC_SIZE			2456
0B		51	91	00131	18\$:	CMPB	R1	#11		2457
		04	12	00134		BNEQ	19\$			
AC	AD	16	B0	00136		MOVW	#22, ACC_SIZE			2459
0B		54	E9	0013A	19\$:	BLBC	R4, 21\$			2465
	AC	AD	B6	0013D		INCW	ACC_SIZE			2467
		06	11	00140		BRB	21\$			2340
AC	AD	03F2	8F	B0	00142	20\$:	MOVW	#1010, ACC_SIZE		2473



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$ACC\_SCR - ACCEPT with screen enhancements

K 4  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 38  
(4)

50	0398	50	AC	AD	B0	00148	21\$:	MOVW	ACC_SIZE, PUT_SIZE	2481	
		8F	AC	AD	B1	0014C		CMPW	ACC_SIZE, #920	2482	
				05	1E	00152		BGEQU	22\$		
50	AC	AD		05	A1	00154		ADDW3	#5, ACC_SIZE, PUT_SIZE		
	20	AE		94	AD	9F	00159	22\$:	PUSHAB	PUT_HERE	2484
				50	3C	0015C		MOVZWL	PUT_SIZE, 32(SP)		
		00		20	AE	9F	00160		PUSHAB	32(SP)	
	00000000G	00		02	FB	00163		CALLS	#2, STR\$GET1_DX		
		0D		50	E8	0016A		BLBS	R0, 23\$		
			00000000G	8F	DD	0016D		PUSHL	#COB\$ ERRDURACC	2485	
	00000000G	00		01	FB	00173		CALLS	#1, LIB\$STOP		
		52		04	AC	9A	0017A	23\$:	MOVZBL	UNIT, R2	2493
		06		52	91	0017E		CMPB	R2, #6		
				0D	1B	00181		BLEQU	24\$		
			00000000G	8F	DD	00183		PUSHL	#COB\$ INVARG	2495	
	00000000G	00		01	FB	00189		CALLS	#1, LIB\$STOP		
	14	AE	00000000G	042	DE	00190	24\$:	MOVAL	COB\$AL_WRITE_RAB[R2], 20(SP)	2497	
			14	BE	D5	00199		TSTL	@20(SP)		
				11	12	0019C		BNEQ	27\$		
04		59		0B	E1	0019E		BBC	#11, R9, 25\$	2504	
				01	DD	001A2		PUSHL	#1		
				02	11	001A4		BRB	26\$		
				7E	D4	001A6	25\$:	CLRL	-(SP)		
				52	DD	001A8	26\$:	PUSHL	R2	2503	
	0000V	CF		02	FB	001AA		CALLS	#2, COB\$OPEN_IN		
		57	14	BE	D0	001AF	27\$:	MOVL	@20(SP), RAB	2508	
		50	44	A7	9E	001B3		MOVAB	68(R7), NAM_DSC	2519	
			00000000G	00	D5	001B7		TSTL	COB\$ACC_TERM_TYPE	2521	
				23	12	001BD		BNEQ	28\$		
			00000000G	00	9F	001BF		PUSHAB	COB\$ACC_TERM_TYPE	2523	
		7E		60	3C	001C5		MOVZWL	(NAM_DSC), -(SP)	2524	
			04	A0	DD	001C8		PUSHL	4(NAM_DSC)	2523	
	00000000G	00		03	FB	001CB		CALLS	#3, COB\$SETUP_TERM_TYPE		
		0D		50	E8	001D2		BLBS	R0, 28\$		
			00000000G	8F	DD	001D5		PUSHL	#COB\$ ERRDURACC	2526	
	00000000G	00		01	FB	001DB		CALLS	#1, LIB\$STOP		
			00000000G	00	D5	001E2	28\$:	TSTL	COB\$ACC_TERM_TYPE	2528	
				46	12	001E8		BNEQ	31\$		
				6E	DD	001EA		PUSHL	YES_SIGN	2539	
				0D	AD	001EC		PUSHL	YES_PROTECT		
				10	AE	001EF		PUSHL	YES_CONV	2538	
				94	AD	9F	001F2	PUSHAB	PUT_HERE	2537	
		7E		AC	AD	3C	001F5	MOVZWL	ACC_SIZE, -(SP)	2538	
				1C	AC	DD	001F9	PUSHL	LENGTH		
				10	AC	DD	001FC	PUSHL	DEFAULT	2537	
				59	DD	001FF		PUSHL	R9		
		7E	04	AC	7D	00201		MOVQ	UNIT, -(SP)		
	0000V	CF		0A	FB	00205		CALLS	#10, COB\$ACC_SCR_FILE		
		52		50	D0	0020A		MOVL	R0, STATUS		
			94	AD	9F	0020D		PUSHAB	PUT_HERE	2543	
	00000000G	00		01	FB	00210		CALLS	#1, STR\$FREE1_DX		
		0D		50	E8	00217		BLBS	R0, 29\$		
			00000000G	8F	DD	0021A		PUSHL	#COB\$ ERRDURACC	2544	
	00000000G	00		01	FB	00220		CALLS	#1, LIB\$STOP		
		03		52	E8	00227	29\$:	BLBS	STATUS, 30\$	2546	
				0542	31	0022A		BRW	106\$		
				053B	31	0022D	30\$:	BRW	105\$		



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COBSACC\_SCR - ACCEPT with screen enhancements

15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 39  
(4)

		00000000'	EF	01	D0	00230	31\$:	MOVL	#1, ACC_SCR	:	2560	
		50	00000000G	00	9A	00237		MOVZBL	COB\$\$AB_PREV, R0	:	2575	
				0A	13	0023E		BEQL	32\$	:		
		02		50	91	00240		CMPB	R0, #2	:	2576	
				05	13	00243		BEQL	32\$	:		
		04		50	91	00245		CMPB	R0, #4	:	2577	
				09	12	00248		BNEQ	33\$	:		
				59	DD	0024A	32\$:	PUSHL	R9	:	2582	
				01	DD	0024C		PUSHL	#1	:		
		0000V	CF	02	FB	0024E		CALLS	#2, COB\$\$RMS_PUT_BYTE	:		
D8	AD	59	04	00	EF	00253	33\$:	EXTZV	#0, #4, R9, PUT_FLAG	:	2591	
				2C	13	00259		BEQL	34\$	:	2593	
				E8	AD	9F	0025B	PUSHAB	OFF_LEN	:	2597	
				DC	AD	9F	0025E	PUSHAB	OFF_BUF	:		
				28	AE	9F	00261	PUSHAB	ON_LEN	:	2596	
				EC	AD	9F	00264	PUSHAB	ON_BUF	:		
				D8	AD	DD	00267	PUSHL	PUT_FLAG	:	2595	
		00000000G	00	00	DD	0026A		PUSHL	COB\$ACC_TERM_TYPE	:		
			0D	06	FB	00270		CALLS	#6, COB\$\$SET_ATTRIBUTES_ONLY	:		
		00000000G	00	50	E8	00277		BLBS	R0, 34\$	:		
				8F	DD	0027A		PUSHL	#COB\$ERRDURACC	:	2598	
		00000000G	00	01	FB	00280		CALLS	#1, LIB\$STOP	:		
OC			59	04	E1	00287	34\$:	BBC	#4, R9, 35\$	:	2604	
			50	EC	AD	9E	0028B	MOVAB	ON_BUF, R0	:	2607	
		20	BE40	07	90	0028F		MOVB	#7, @ON_LEN[R0]	:		
				20	AE	D6	00294	INCL	ON_LEN	:	2608	
				06	6C	91	00297	CMPB	(AP), #6	:	2618	
					58	1F	0029A	BLSSU	41\$	:		
				18	AC	D5	0029C	TSTL	24(AP)	:		
					53	13	0029F	BEQL	41\$	:		
		28	AE	18	BC	3C	002A1	MOVZWL	@KEY, KEY_LEN	:	2624	
				24	AE	9F	002A6	PUSHAB	BLANKS	:	2625	
				2C	AE	9F	002A9	PUSHAB	KEY_LEN	:		
				18	AC	DD	002AC	PUSHL	KEY	:		
		00000000G	00	03	FB	002AF		CALLS	#3, STR\$DUPL_CHAR	:		
			07	6C	91	002B6		CMPB	(AP), #7	:	2632	
				05	1F	002B9		BLSSU	36\$	:		
				1C	AC	D5	002BB	TSTL	28(AP)	:		
					34	12	002BE	BNEQ	41\$	:		
			05	6C	91	002C0	36\$:	CMPB	(AP), #5	:	2633	
				05	1F	002C3		BLSSU	37\$	:		
				14	AC	D5	002C5	TSTL	20(AP)	:		
					2A	12	002C8	BNEQ	41\$	:		
			04	6C	91	002CA	37\$:	CMPB	(AP), #4	:	2634	
				05	1F	002CD		BLSSU	38\$	:		
				10	AC	D5	002CF	TSTL	16(AP)	:		
					20	12	002D2	BNEQ	41\$	:		
			02	6C	91	002D4	38\$:	CMPB	(AP), #2	:	2635	
				05	1F	002D7		BLSSU	39\$	:		
				08	AC	D5	002D9	TSTL	8(AP)	:		
					16	12	002DC	BNEQ	41\$	:		
				18	AC	DD	002DE	39\$:	PUSHL	KEY	:	2637
					59	DD	002E1	PUSHL	R9	:		
				04	AC	DD	002E3	PUSHL	UNIT	:		
		0000V	CF	03	03	FB	002E6	CALLS	#3, COB\$\$FORMAT_FOUR	:		
					50	E8	002EB	BLBS	R0, 40\$	:		
				047E	31	002EE		BRW	106\$	:		



				0477	31	002F1	40\$:	BRW	105\$			
			18	AE	D4	002F4	41\$:	CLRL	24(SP)		2653	
	OF		59	09	E1	002F7		BBC	#9, R9, 42\$			
			18	AE	D6	002FB		INCL	24(SP)			
		B4	AD	5240	8F	3C	002FE	MOVZWL	#21056, FUNC_VAL		2657	
		04	AE		01	D0	00304	MOVL	#1, YES_NO_ECHO		2658	
					06	11	00308	BRB	43\$		2653	
		B4	AD	5200	8F	3C	0030A	MOVZWL	#20992, FUNC_VAL		2661	
					5A	D5	00310	43\$:	TSTL	PROT_OK	2671	
					08	13	00312	BEQL	44\$			
			10	AE	D5	00314		TSTL	CONV_OK			
				03	13	00317		BEQL	44\$			
				041C	31	00319		BRW	103\$			
				58	D4	0031C	44\$:	CLRL	TERM_SEEN		2672	
			0C	AE	D5	0031E		TSTL	REPRMPT_DONE		2676	
				03	13	00321		BEQL	45\$			
				008D	31	00323		BRW	49\$			
			03	00000000G	00	D1	00326	45\$:	CML	COBSACC_TERM_TYPE, #3	2688	
				55	12	0032D		BNEQ	47\$			
			51	D0	AD	E9	0032F	BLBC	YES_PROTECT, 47\$		2691	
	FE68	CD	EC	AD	20	AE	28	00333	MOVCS	ON_LEN, ON_BUF, FIELD_BUF	2706	
				56	20	AE	D0	0033B	MOVL	ON_LEN, FIELD_LEN	2707	
AC	AD		20	6E	00	2C	0033F	MOVCS	#0, (SP), #32, ACC_SIZE, FIELD_BUF-		2708	
					FE68	CD46			[FIELD_LEN]			
				50	AC	AD	3C	00349	MOVZWL	ACC_SIZE, RO	2709	
				56	50	C0	0034D	ADDL2	RO, FIELD_LEN			
AC	AD		08	6E	00	2C	00350	MOVCS	#0, (SP), #8, ACC_SIZE, FIELD_BUF-		2710	
					FE68	CD46			[FIELD_LEN]			
				50	AC	AD	3C	0035A	MOVZWL	ACC_SIZE, RO	2711	
				56	50	C0	0035E	ADDL2	RO, FIELD_LEN			
		000003F2		8F	56	D1	00361	CML	FIELD_LEN, #1010		2721	
					0D	15	00368	BLEQ	46\$			
				00000000G	8F	DD	0036A	PUSHL	#COBS_ERRDURACC		2723	
			00		01	FB	00370	CALLS	#1, LIB\$STOP			
				0240	8F	BB	00377	46\$:	PUSHR	#M<R6,R9>	2729	
				FE68	CD	9F	0037B		PUSHAB	FIELD_BUF		
		0000V	CF		03	FB	0037F	CALLS	#3, COB\$SRMS_PUT_BUFFER			
				20	AE	D5	00384	47\$:	TSTL	ON_LEN	2749	
					13	13	00387	BEQL	48\$			
			01	D0	AD	D1	00389	CML	YES_PROTECT, #1			
					0D	13	0038D	BEQL	48\$			
					59	DD	0038F	PUSHL	R9		2751	
				24	AE	DD	00391	PUSHL	ON_LEN			
				EC	AD	9F	00394	PUSHAB	ON_BUF			
		0000V	CF		03	FB	00397	CALLS	#3, COB\$SRMS_PUT_BUFFER			
				57	14	BE	D0	0039C	48\$:	MOVL	@20(SP), RAB	2757
					98	AD	DD	003A0	PUSHL	PUT_HERE+4	2759	
			7E	AC	AD	3C	003A3	MOVZWL	ACC_SIZE, -(SP)		2758	
				B4	AD	DD	003A7	PUSHL	FUNC_VAL			
					57	DD	003AA	PUSHL	RAB			
		0000V	CF		04	FB	003AC	CALLS	#4, COB\$SRMS_GET			
					03	11	003B1	BRB	50\$		2676	
				0C	AE	D4	003B3	49\$:	CLRL	REPRMPT_DONE	2763	
				22	A7	3C	003B6	50\$:	MOVZWL	34(RAB), CHARS_READ	2776	
		B0	AD		0E	A7	9A	003BB	MOVZBL	14(RAB), TERM_SIZE	2777	
		B8	AD		0C	A7	9A	003C0	MOVZBL	12(RAB), TERM_LOC	2778	
		BC	AD		08	A7	D1	003C5	CML	8(RAB), #98760	2787	
		000181C8	8F									



			0E 12 003CD	BNEQ	51\$		
	58		01 D0 003CF	MOVL	#1, TERM_SEEN		2790
		04	AC DD 003D2	PUSHL	UNIT		2791
		94	AD 9F 003D5	PUSHAB	PARAMETERS		
0000V	CF		02 FB 003D8	CALLS	#2, COB\$\$\$PARTIAL_SEQ		
7F	8F	0C	A7 91 003DD	CMPB	12(RAB), #127		2798
			0D 12 003E2	BNEQ	52\$		
			59 DD 003E4	PUSHL	R9		2800
		04	AC DD 003E6	PUSHL	UNIT		
		94	AD 9F 003E9	PUSHAB	PARAMETERS		
0000V	CF		03 FB 003EC	CALLS	#3, COB\$\$\$DELETE_KEY		
	03	D0	AD E8 003F1	BLBS	YES_PROTECT, 54\$		2825
		00A8	31 003F5	BRW	63\$		
		B0	AD D5 003F8	TSTL	CHARS_READ		2826
			F8 13 003FB	BEQL	53\$		
000181B8	8F	08	A7 D1 003FD	CMPL	8(RAB), #98744		2828
			EE 12 00405	BNEQ	53\$		
			58 D5 00407	TSTL	TERM_SEEN		2829
			EA 12 00409	BNEQ	53\$		
			53 D4 0040B	CLRL	NO_CHAR		2831
			52 D4 0040D	CLRL	HAVE_TERM		
	01		52 D1 0040F	CMPL	HAVE_TERM, #1		2850
			03 12 00412	BNEQ	57\$		
		008C	31 00414	BRW	64\$		
			53 D4 00417	CLRL	NO_CHAR		2853
1C	AE	5240	8F 3C 00419	MOVZWL	#2T056, FUNC_VAL_2		2855
	57	14	BE D0 0041F	MOVL	@20(SP), RAB		2857
		A0	AD 9F 00423	PUSHAB	NEXT_CHAR		2858
			01 DD 00426	PUSHL	#1		
		24	AE DD 00428	PUSHL	FUNC_VAL_2		
			57 DD 0042B	PUSHL	RAB		
0000V	CF		04 FB 0042D	CALLS	#4, COB\$\$\$RMS_GET		
		22	A7 B5 00432	TSTW	34(RAB)		2868
			08 12 00435	BNEQ	58\$		
	53		01 D0 00437	MOVL	#1, NO_CHAR		2871
A0	AD	0C	A7 90 0043A	MOVB	12(RAB), NEXT_CHAR		2872
B8	AD	0E	A7 9A 0043F	MOVZBL	14(RAB), TERM_SIZE		2874
BC	AD	0C	A7 9A 00444	MOVZBL	12(RAB), TERM_LOC		2875
C4	AD		01 D0 00449	MOVL	#1, TERM_IN_NEXT		2876
000181C8	8F	08	A7 D1 0044D	CMPL	8(RAB), #98760		2877
			0B 12 00455	BNEQ	59\$		
		04	AC DD 00457	PUSHL	UNIT		2879
		94	AD 9F 0045A	PUSHAB	PARAMETERS		
0000V	CF		02 FB 0045D	CALLS	#2, COB\$\$\$PARTIAL_SEQ		
	2D		53 E9 00462	BLBC	NO_CHAR, 62\$		2886
	5A		01 D0 00465	MOVL	#1, PROT_OK		2889
	52		01 D0 00468	MOVL	#1, HAVE_TERM		2890
7F	8F	0C	A7 91 0046B	CMPB	12(RAB), #127		2901
			9D 12 00470	BNEQ	56\$		
			59 DD 00472	PUSHL	R9		2904
		04	AC DD 00474	PUSHL	UNIT		
		94	AD 9F 00477	PUSHAB	PARAMETERS		
0000V	CF		03 FB 0047A	CALLS	#3, COB\$\$\$DELETE_KEY		
		B8	AD D5 0047F	TSTL	TERM_SIZE		2910
			06 12 00482	BNEQ	61\$		
			52 D4 00484	CLRL	HAVE_TERM		2913
			5A D4 00486	CLRL	PROT_OK		2914



			85	11	00488	60\$:	BRB	56\$		2910
		52	01	D0	0048A	61\$:	MOVL	#1, HAVE_TERM		2918
			C4	AD	D4	0048D	CLRL	TERM_IN_NEXT		2919
				F6	11	00490	BRB	60\$		2901
				59	DD	00492	PUSHL	R9		2939
				02	DD	00494	PUSHL	#2		
	0000V	CF		02	FB	00496	CALLS	#2, COB\$SRMS_PUT_BYTE		
				5A	D4	0049B	CLRL	PROT_OK		2940
			FF	6D	31	0049D	BRW	55\$		2941
		5A		01	D0	004A0	MOVL	#1, PROT_OK		2956
		6B		5A	E9	004A3	BLBC	PROT_OK, -70\$		2962
		07	C4	AD	E9	004A6	BLBC	TERM_IN_NEXT, 65\$		2969
	CO	AD	A0	AD	9E	004AA	MOVAB	NEXT_CHAR, TERM_PTR		2971
				07	11	004AF	BRB	66\$		
CO	AD		B0	AD	C1	004B1	ADDL3	CHARS_READ, PUT_HERE+4, TERM_PTR		2973
		98	B8	AD	D1	004B8	CMPL	TERM_SIZE, #1		2990
		01		5D	12	004BC	BNEQ	72\$		
	CO	AD	OC	A7	9E	004BE	MOVAB	12(RAB), TERM_PTR		2993
		50	OC	A7	9A	004C3	MOVZBL	12(RAB), R0		2994
		09		50	91	004C7	CMPB	R0, #9		2996
				05	13	004CA	BEQL	67\$		
		0D		50	91	004CC	CMPB	R0, #13		
				15	12	004CF	BNEQ	68\$		
		06		6C	91	004D1	CMPB	(AP), #6		3001
				3B	1F	004D4	BLSSU	70\$		
			18	AC	D5	004D6	TSTL	24(AP)		
				36	13	004D9	BEQL	70\$		
		50	18	AC	D0	004DB	MOVL	KEY, R0		3003
	04	B0	CO	BD	90	004DF	MOVAB	@TERM_PTR, @4(R0)		
				2B	11	004E4	BRB	70\$		3001
		1A		50	91	004E6	CMPB	R0, #26		3005
				13	12	004E9	BNEQ	69\$		
24		59		0B	E0	004EB	BBS	#11, R9, 71\$		3013
				59	DD	004EF	PUSHL	R9		3028
			94	AD	9F	004F1	PUSHAB	PARAMETERS		
	0000V	CF		02	FB	004F4	CALLS	#2, COB\$SCLEAN_UP		
			18	AC	DD	004F9	PUSHL	KEY		3029
				40	11	004FC	BRB	73\$		
	7F	8F		50	91	004FE	CMPB	R0, #127		3034
				0F	12	00502	BNEQ	71\$		
				59	DD	00504	PUSHL	R9		3037
			04	AC	DD	00506	PUSHL	UNIT		
			94	AD	9F	00509	PUSHAB	PARAMETERS		
	0000V	CF		03	FB	0050C	CALLS	#3, COB\$DELETE_KEY		
				64	11	00511	BRB	77\$		2994
			CC	AD	D4	00513	CLRL	LEGAL		3043
			18	AC	DD	00516	PUSHL	KEY		3045
				4F	11	00519	BRB	76\$		3044
		52	18	AC	D0	0051B	MOVL	KEY, R2		3062
			B0	AD	D5	0051F	TSTL	CHARS_READ		3051
				25	12	00522	BNEQ	74\$		
	0001827A	8F	08	A7	D1	00524	CMPL	8(RAB), #98938		
				1B	12	0052C	BNEQ	74\$		
33		59		0B	E0	0052E	BBS	#11, R9, 75\$		3058
				59	DD	00532	PUSHL	R9		3066
			94	AD	9F	00534	PUSHAB	PARAMETERS		
	0000V	CF		02	FB	00537	CALLS	#2, COB\$SCLEAN_UP		



		04	52	DD	0053C	PUSHL	R2		3067
			AC	DD	0053E	PUSHL	UNIT		
0000V	CF		02	FB	00541	CALLS	#2, COB\$\$\$CONTROL_Z		
		0226	31		00546	BRW	106\$		3068
	06		6C	91	00549	CMPB	(AP), #6		3079
			17	1F	0054C	BLSSU	75\$		
		18	AC	D5	0054E	TSTL	24(AP)		
			12	13	00551	BEQL	75\$		
			52	DD	00553	PUSHL	R2		3087
		B8	AD	DD	00555	PUSHL	TERM_SIZE		
		CO	AD	9F	00558	PUSHAB	TERM_PTR		
00000000G	00		03	FB	0055B	CALLS	#3, COB\$\$\$CONTROL_KEY		
	12		50	E8	00562	BLBS	R0, 77\$		
		CC	AD	D4	00565	CLRL	LEGAL		3101
			52	DD	00568	PUSHL	R2		3102
			59	DD	0056A	PUSHL	R9		
		04	AC	DD	0056C	PUSHL	UNIT		
0000V	CF	94	AD	9F	0056F	PUSHAB	PARAMETERS		
			04	FB	00572	CALLS	#4, COB\$\$\$ILLEGAL_TERM		
			52	D4	00577	CLRL	R2		3123
		B0	AD	D5	00579	TSTL	CHARS_READ		
			10	12	0057C	BNEQ	78\$		
			52	D6	0057E	INCL	R2		
0A	59		0B	E1	00580	BBC	#11, R9, 78\$		
			59	DD	00584	PUSHL	R9		3130
0000V	CF		01	FB	00586	CALLS	#1, COB\$\$\$RPG_CLEAN_UP		
		01DD	31		0058B	BRW	105\$		3131
	45		52	E9	0058E	BLBC	R2, 80\$		3134
	04		6C	91	00591	CMPB	(AP), #4		3136
			40	1F	00594	BLSSU	80\$		
		10	AC	D5	00596	TSTL	16(AP)		
			3B	13	00599	BEQL	80\$		
		D4	AD	D5	0059B	TSTL	YES_DEFAULT		
			36	12	0059E	BNEQ	80\$		
B0	AD	10	BC	3C	005A0	MOVZWL	@DEFAULT, CHARS_READ		3140
D4	AD		01	D0	005A5	MOVL	#1, YES_DEFAULT		3141
	26	D0	AD	E9	005A9	BLBC	YES_PROTECT, 79\$		3149
	50	08	AC	D0	005AD	MOVL	STRING_DEST, R0		3150
	0A	02	A0	91	005B1	CMPB	2(R0), #10		
			1C	13	005B5	BEQL	79\$		
	0B	02	A0	91	005B7	CMPB	2(R0), #11		3151
			16	13	005BB	BEQL	79\$		
AC	AD	10	BC	B1	005BD	CMPW	@DEFAULT, ACC_SIZE		3161
			0F	1B	005C2	BLEQU	79\$		
00000000G	00	00000000G	8F	DD	005C4	PUSHL	#COB\$ INVDEFVAL		3163
			01	FB	005CA	CALLS	#1, LIB\$STOP		
			03	11	005D1	BRB	80\$		
	5A		01	D0	005D3	MOVL	#1, PROT_OK		3167
	5A		5A	E9	005D6	BLBC	PROT_OK, 86\$		3180
	20	08	AE	E9	005D9	BLBC	YES_CONV, 81\$		3182
			6E	DD	005DD	PUSHL	YES_SIGN		3186
		D4	AD	DD	005DF	PUSHL	YES_DEFAULT		
B0	AD		AD	DD	005E2	PUSHL	CHARS_READ		3185
94	AD		9F	005E5	PUSHAB	PUT_HERE			3184
		10	AC	DD	005E8	PUSHL	DEFAULT		3185
			59	DD	005EB	PUSHL	R9		3184
		08	AC	DD	005ED	PUSHL	STRING_DEST		



			00000000G	00	07	FB	005F0	CALLS	#7, COB\$ACC_CONVERT		
			10	AE	50	DO	005F7	MOVL	R0, CONV_OK		
B0	AD				36	11	005FB	BRB	86\$		
		08	BC	10	00	ED	005FD	CMPZV	#0, #16, @STRING_DEST, CHARS_READ	3196	
					07	15	00604	BLEQ	82\$		
			2C	AE	B0	AD	DO	00606	MOVL	CHARS_READ, COPY_NUM	3198
					05	11	0060B	BRB	83\$		
			2C	AE	08	BC	3C	0060D	MOVZWL	@STRING_DEST, COPY_NUM	3200
				09	D4	AD	E9	00612	BLBC	YES_DEFAULT, 84\$	3203
				50	10	AC	DO	00616	MOVL	DEFAULT, R0	3204
					04	A0	DD	0061A	PUSHL	4(R0)	
					03	11	0061D	BRB	85\$		
					98	AD	DD	0061F	PUSHL	PUT_HERE+4	3205
					30	AE	9F	00622	PUSHAB	COPY_NUM	3202
					08	AC	DD	00625	PUSHL	STRING_DEST	
			00000000G	00	03	FB	00628	CALLS	#3, STR\$COPY_R		
			10	AE	01	DO	0062F	MOVL	#1, CONV_OK	3208	
					10	AE	D5	00633	TSTL	CONV_OK	3215
					03	13	00636	BEQL	87\$		
					00F6	31	00638	BRW	101\$		
		12		59	0B	E1	0063B	BBC	#11, R9, 88\$	3229	
					59	DD	0063F	PUSHL	R9	3236	
					02	DD	00641	PUSHL	#2		
			0000V	CF	02	FB	00643	CALLS	#2, COB\$RMS_PUT_BYTE		
					59	DD	00648	PUSHL	R9	3237	
			0000V	CF	01	FB	0064A	CALLS	#1, COB\$RPG_CLEAN_UP		
					10	11	0064F	BRB	89\$	3238	
				02	05	AC	91	00651	CMPB	UNIT+1, #2	3241
					0D	12	00655	BNEQ	90\$		
					59	DD	00657	PUSHL	R9	3247	
					94	AD	9F	00659	PUSHAB	PARAMETERS	
			0000V	CF	02	FB	0065C	CALLS	#2, COB\$SCLEAN_UP		
					010B	31	00661	BRW	106\$	3248	
				04	18	AE	E8	00664	BLBS	24(SP), 91\$	3258
				0C	D4	AD	E9	00668	BLBC	YES_DEFAULT, 92\$	
					59	DD	0066C	PUSHL	R9	3261	
					02	DD	0066E	PUSHL	#2		
			0000V	CF	02	FB	00670	CALLS	#2, COB\$RMS_PUT_BYTE		
					00B1	31	00675	BRW	100\$	3258	
					56	D4	00678	CLRL	PUT_TOTAL	3264	
					58	D4	0067A	CLRL	INDEX		
					59	DD	0067C	PUSHL	R9		
					02	DD	0067E	PUSHL	#2		
			0000V	CF	02	FB	00680	CALLS	#2, COB\$RMS_PUT_BYTE		
					04	AE	D5	00685	TSTL	YES_NO_ECHO	
					59	12	00688	BNEQ	96\$		
					5B	D4	0068A	CLRL	R11		
					D8	AD	D5	0068C	TSTL	PUT_FLAG	
					12	13	0068F	BEQL	93\$		
					5B	D6	00691	INCL	R11		
					D0	AD	D5	00693	TSTL	YES_PROTECT	
					0B	12	00696	BNEQ	93\$		
		30	AE	DC	E8	AD	28	00698	MOVC3	OFF_LEN, OFF_BUF, RESTORE_CURSOR	
					E8	AD	D0	0069F	MOVL	OFF_LEN, PUT_TOTAL	
					56	D0	006A3	MOVL	PUT_TOTAL, INDEX		
		51		56	B0	AD	C1	006A6	ADDL3	CHARS_READ, PUT_TOTAL, R1	
				50	FF	A6	9E	006AB	MOVAB	-1(R6), P	



			12	11	006AF		BRB	95\$		
	30	AE48	08	90	006B1	94\$:	MOVB	#8, RESTORE_CURSOR[INDEX]		
	31	AE48	20	90	006B6		MOVB	#32, RESTORE_CURSOR+1[INDEX]		
	32	AE48	08	90	006BB		MOVB	#8, RESTORE_CURSOR+2[INDEX]		
		58	03	C0	006C0		ADDL2	#3, INDEX		
EA		50	51	F2	006C3	95\$:	AOBLSS	R1, P, 94\$		
50			03	C5	006C7		MULL3	#3, CHARS_READ, R0		
	B0	AD	50	C0	006CC		ADDL2	R0, PUT_TOTAL		
		56	5B	E9	006CF		BLBC	R11, 96\$		
		11					TSTL	YES_PROTECT		
			D0	AD	D5 006D2		BNEQ	96\$		
				0C	12 006D5		MOVC3	ON_LEN, ON_BUF, RESTORE_CURSOR[PUT_TOTAL]		
30	AE46		20	AE	28 006D7		ADDL2	ON_LEN, PUT_TOTAL		
		EC	20	AE	C0 006DF		CLRL	LAST_WRITE		
				51	D4 006E3	96\$:	TSTL	LAST_WRITE		
				51	D5 006E5	97\$:	BNEQ	99\$		
				1B	12 006E7		CMP	PUT_TOTAL, #1010		
	000003F2	8F		56	D1 006E9		BLEQ	98\$		
				0A	15 006F0		MOVZWL	#1010, P_TOT		
		50	03F2	8F	3C 006F2		SUBL2	P_TOT, PUT_TOTAL		
		56		50	C2 006F7		BRB	97\$		
				E9	11 006FA		MOVL	PUT_TOTAL, P_TOT		
		50		56	D0 006FC	98\$:	MOVL	#1, LAST_WRITE		
		51		01	D0 006FF		BRB	97\$		
				E1	11 00702		PUSHR	#*M<R0,R9>		
			0201	8F	BB 00704	99\$:	PUSHAB	RESTORE_CURSOR		
			38	AE	9F 00708		CALLS	#3, COB\$SRMS_PUT_BUFFER		
	0000V	CF		03	FB 0070B		MOVL	@20(SP), RAB		
		57	14	BE	D0 00710		PUSHL	PUT_HERE+4		
			98	AD	DD 00714		MOVZWL	ACC_SIZE, -(SP)		
		7E		AC	3C 00717		PUSHL	FUNC_VAL		
				B4	AD DD 0071B		PUSHL	RAB		
				57	DD 0071E		CALLS	#4, COB\$SRMS_GET		
	0000V	CF		04	FB 00720		MOVL	#1, REPROMPT_DONE		
	OC	AE		01	D0 00725		CLRL	CONV_OK		3267
			10	AE	D4 00729	100\$:	CLRL	YES_DEFAULT		3268
			D4	AD	D4 0072C		BRB	102\$		3215
				04	11 0072F		MOVL	#1, CONV_OK		3277
	10	AE		01	D0 00731	101\$:	BRW	43\$		2671
				FB08	31 00735	102\$:	CMPB	(AP), #7		3286
		07		6C	91 00738	103\$:	BLSSU	104\$		
				0A	1F 0073B		TSTL	28(AP)		
			1C	AC	D5 0073D		BEQL	104\$		
				05	13 00740		MOVL	CHARS_READ, @LENGTH		3288
	1C	BC	B0	AD	D0 00742		PUSHL	R9		3299
				59	DD 00747	104\$:	PUSHAB	PARAMETERS		
			94	AD	9F 00749		CALLS	#2, COB\$SCLEAN_UP		
	0000V	CF		02	FB 0074C		PUSHAB	PUT_HERE		3307
			94	AD	9F 00751		CALLS	#1, STR\$FREE1_DX		
	00000000G	00		01	FB 00754		BLBS	R0, 105\$		
		OD		50	E8 0075B		PUSHL	#COB\$ERRDURACC		3308
			00000000G	8F	DD 0075E		CALLS	#1, LIB\$STOP		3311
	00000000G	00		01	FB 00764	105\$:	MOVL	#1, R0		
		50		01	D0 0076B	106\$:	RET	R0		3312
				04	0076E		CLRL			
				50	D4 0076F		RET			
				04	00771					



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$ACC\_SCR - ACCEPT with screen enhancements

F 5  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 46  
(4)

; Routine Size: 1906 bytes,      Routine Base: \_COB\$CODE + 0238



```

: 1801 3313 1 %SBTTL 'COBSACC_SCR_FILE - Screen enhancements for files'
: 1802 3314 1 ROUTINE COBSACC_SCR_FILE ( UNIT      : VECTOR [2, BYTE],
: 1803 3315 1                                     STRING_DEST : REF $STR$DESCRIPTOR,
: 1804 3316 1                                     FLAGS,
: 1805 3317 1                                     DEFAULT      : REF $STR$DESCRIPTOR,
: 1806 3318 1                                     LENGTH,
: 1807 3319 1                                     ACC_SIZE,
: 1808 3320 1                                     PUT_HERE     : REF BLOCK [8, BYTE],
: 1809 3321 1                                     | Contains input characters
: 1810 3322 1                                     | =1 if conversion requested
: 1811 3323 1                                     | =1 if protection requested
: 1812 3324 1                                     | =1 if sign should be included
: 1813 3325 1                                     ) =
: 1814 3326 1
: 1815 3327 1 ++
: 1816 3328 1 FUNCTIONAL DESCRIPTION:
: 1817 3329 1
: 1818 3330 1 This routine handles the VAX COBOL Version 3 ACCEPT statement
: 1819 3331 1 with Screen Enhancements when a file (not a terminal) is used
: 1820 3332 1 for input. A non terminal $GET service does not contain all the
: 1821 3333 1 features of a terminal $GET service, so this routine is a scaled
: 1822 3334 1 down version of COBSACC_SCR. Note that the fields RAB [RAB$V_ETO]
: 1823 3335 1 and RAB [RAB$L_XAB] are not set.
: 1824 3336 1
: 1825 3337 1 FORMAL PARAMETERS:
: 1826 3338 1
: 1827 3339 1 UNIT.rbu.va Array of two unsigned byte integers.
: 1828 3340 1 The first byte is the unit number designating the
: 1829 3341 1 device from which the string is to be read.
: 1830 3342 1 The second byte indicates whether the routine should
: 1831 3343 1 abort or return to the calling program.
: 1832 3344 1 Byte 2 = 0 - routine will abort on control z
: 1833 3345 1 and reprompt on conversion errors.
: 1834 3346 1 = 1 - ( AT END )
: 1835 3347 1 routine will return to calling program
: 1836 3348 1 on control z and reprompt on conversion
: 1837 3349 1 errors.
: 1838 3350 1 = 2 - ( ON EXCEPTION )
: 1839 3351 1 routine will return to calling program
: 1840 3352 1 on control z and conversion errors.
: 1841 3353 1
: 1842 3354 1 STRING_DEST.mt.ds Address of descriptor to receive the read input.
: 1843 3355 1
: 1844 3356 1 FLAGS.rlu.v Screen enhancement flag;
: 1845 3357 1
: 1846 3358 1 bit 0 - bold
: 1847 3359 1 bit 1 - reverse
: 1848 3360 1 bit 2 - blink
: 1849 3361 1 bit 3 - underline
: 1850 3362 1 bit 4 - bell
: 1851 3363 1 bit 5 - conversion
: 1852 3364 1 bit 6 - decimal point is comma
: 1853 3365 1 bit 7 - 0 to allow space for sign in PROTECTED
: 1854 3366 1 ACCEPT, 1 no allowance for sign
: 1855 3367 1 bit 8 - protect
: 1856 3368 1 bit 9 - no-echo
: 1857 3369 1 bit 10 - 0 advancing, 1 no advancing
```



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement

COBSACC\_SCR\_FILE - Screen enhancements for file

H 5

15-Sep-1984 23:54:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 48  
(5)

bit 11 - 0 for VAX COBOL, 1 for VAX RPG

DEFAULT.rt.dx Default source moved to destination descriptor  
(STRING\_DEST) in the event of null input.

LENGTH.wlu.r Destination of the number of characters read.

ACC\_SIZE.rlu.v # of characters to RMS \$GET.

PUT\_HERE.rt.dx Buffer to hold input characters.

YES\_CONV.rlu.v Flag = 1 if Conversion requested by user.

YES\_PROTECT.rlu.v Flag = 1 if Protection requested by user.

YES\_SIGN.rlu.v Flag = 1 if sign should be included in COMP or COMP3  
data type.

IMPLICIT INPUTS:

Status of whether the input file is currently open.

IMPLICIT OUTPUTS:

Updated status of file

ROUTINE VALUE:

If .UNIT[1] is false : Unspecified.

If .UNIT[1] is true : Either true or false, indicating success or  
EOF, respectively.

SIDE EFFECTS:

Reads a record from a designated uint.

--

BEGIN

LOCAL

RAB : REF \$RAB DECL,  
CR\_BUF : VECTOR [T,BYTE],  
CHARS\_READ : INITIAL (0), ! Number of characters read  
CONV\_OK : INITIAL (0), ! = 1 if no conversion errors  
YES\_DEFAULT : INITIAL (0); ! = 1 if DEFAULT was used as input

BUILTIN

NULLPARAMETER ;

!+  
RMS \$PUT - If previous call requires advancing, \$PUT a linefeed to  
SYSS\$OUTPUT. Open SYSS\$OUTPUT if necessary.  
!-

IF (.COB\$\$AB\_PREV[0] EQL DISP  
OR .COB\$\$AB\_PREV[0] EQL POS



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COBSACC\_SCR\_FILE - Screen enhancements for file

1 5  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 44  
(5)

```
: 1915      3427 3      OR .COB$$AB_PREV[0] EQL ACC_ADV )
: 1916      3428 3      THEN
: 1917      3429 3      COB$$RMS_PUT_BYTE ( LINE_FD, .FLAGS ) ;
: 1918      3430 3
: 1919      3431 3      !+
: 1920      3432 3      RMS $GET to accept input from a file.
: 1921      3433 3      !-
: 1922      3434 3
: 1923      3435 3      RAB = .COB$$AL_WRITE_RAB [.UNIT[0]] ;
: 1924      3436 3      RAB [RAB$W_USZ] = .ACC_SIZE ;
: 1925      3437 3      RAB [RAB$L_UBF] = .PUT_HERE [DSC$A_POINTER] ;
: 1926      3438 3      !+
: 1927      3439 3      Turn off RAB [RAB$V_ETO] just in case a 'screen enhancement ACCEPT'
: 1928      3440 3      was performed before this one.
: 1929      3441 3      !-
: 1930      3442 3      RAB [RAB$V_ETO] = 0 ;
: 1931      3443 3
: 1932      3444 3      WHILE $GET (RAB = .RAB) EQL RMSS_RSA DO $WAIT (RAB = .RAB) ;
: 1933      3445 3
: 1934      3446 3      IF NOT .RAB [RAB$L_STS] AND NULLPARAMETER (DEFAULT)
: 1935      3447 3      THEN
: 1936      3448 3          LIB$STOP (( IF .RAB[RAB$L_STS] EQL RMSS_EOF
: 1937      3449 3              THEN
: 1938      3450 3                  !+
: 1939      3451 3                  If ON EXCEPTION or AT END, return to user program.
: 1940      3452 3                  !-
: 1941      3453 3                  IF .UNIT [1] EQL 1 OR .UNIT [1] EQL 2
: 1942      3454 3                  THEN
: 1943      3455 3                      RETURN 0
: 1944      3456 3                  ELSE
: 1945      3457 3                      COB$_EOFON_ACC
: 1946      3458 3                  ELSE
: 1947      3459 3                      COB$_ERRDURACC),
: 1948      3460 3                      1, .RAB + RAB$C_BLN, .RAB [RAB$L_STS], .RAB [RAB$L_STV] ) ;
: 1949      3461 3
: 1950      3462 3      !+
: 1951      3463 3      Put number of characters read from $GET in CHARS_READ.
: 1952      3464 3      Pass this info along to COB$$ACC_CONVERT.
: 1953      3465 3      !-
: 1954      3466 3
: 1955      3467 3      CHARS_READ = .RAB [RAB$W_RSZ] ;
: 1956      3468 3
: 1957      3469 3      !*****
: 1958      3470 3      !***** NULL INPUT
: 1959      3471 3      !*****
: 1960      3472 3
: 1961      3473 3      !+
: 1962      3474 3      Null input.
: 1963      3475 3      Check for DEFAULT parameter - if present prepare to put it through
: 1964      3476 3      Conversion routines by placing DEFAULT in PUT_HERE.
: 1965      3477 3      !-
: 1966      3478 3
: 1967      3479 3      IF ( .CHARS_READ EQL 0 ) AND (( .FLAGS AND V_COB_RPG ) NEQ 0 )
: 1968      3480 3      THEN
: 1969      3481 3          !+
: 1970      3482 3          In case of null input for RPG, simply return (no DEFAULT),
: 1971      3483 3          after setting advancing flag.
```



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COBSACC\_SCR\_FILE - Screen enhancements for file

J 5  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 50  
(5)

```

1972 3484 2      !-
1973 3485 2      BEGIN
1974 3486 2      IF (.FLAGS AND V_ADV) NEQ 0
1975 3487 2      THEN
1976 3488 2          COB$$AB_PREV[0] = ACC_DNA
1977 3489 2      ELSE
1978 3490 2          COB$$AB_PREV[0] = ACC_ADV ;
1979 3491 2      RETURN 1 ;
1980 3492 2      END ;
1981 3493 2
1982 3494 2      !+
1983 3495 2      There can be no PROTECTION check on input when dealing with files as
1984 3496 2      RMS will only read ACC_SIZE characters or less.  If .ACC_SIZE were 4
1985 3497 2      but the record contained "abcdef", only "abcd" will be pulled from the
1986 3498 2      record.  RMS ignores the remaining characters "ef" and goes on to the
1987 3499 2      next record.  However it is possible to perform a PROTECTION check
1988 3500 2      when the DEFAULT value is used.
1989 3501 2      !-
1990 3502 2
1991 3503 2      IF ( .CHARS_READ EQL 0 )
1992 3504 2      THEN
1993 3505 2          BEGIN
1994 3506 2              IF (.DEFAULT NEQ 0) AND (.YES_DEFAULT EQL 0)
1995 3507 2              THEN
1996 3508 2                  BEGIN                                ! Begin YES Default
1997 3509 2
1998 3510 2                  CHARS_READ = .DEFAULT [DSC$W_LENGTH];
1999 3511 2                  YES_DEFAULT = 1 ;
2000 3512 2
2001 3513 2                  !+
2002 3514 2                  Protection check for DEFAULT excluding the Floating
2003 3515 2                  Point data types ( these will be handled in
2004 3516 2                  COB$$VERIFY_FL_RANGE.
2005 3517 2                  !-
2006 3518 2
2007 3519 2                  IF (.YES_PROTECT AND
2008 3520 2                      ( .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_F AND
2009 3521 2                        .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_D ))
2010 3522 2                  THEN
2011 3523 2                      IF (.DEFAULT [DSC$W_LENGTH] GTR .ACC_SIZE)
2012 3524 2                      THEN
2013 3525 2                          !+
2014 3526 2                          If the length of DEFAULT is greater than the
2015 3527 2                          expected input size ACC_SIZE, then there is a
2016 3528 2                          Protection error.
2017 3529 2                          !-
2018 3530 2                          LIB$STOP ( COB$_INVDEFVAL ) ;
2019 3531 2
2020 3532 2                  END ;                                ! End YES Default
2021 3533 2              END ;
2022 3534 2
2023 3535 2      !*****
2024 3536 2      !***** CONVERSION
2025 3537 2      !*****
2026 3538 2
2027 3539 2      !+
2028 3540 2      ! If conversion requested, call routine COB$$ACC_CONVERT
```



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement

COBSACC\_SCR\_FILE - Screen enhancements for file

K 5  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 51  
(5)

```
2029 3541 2  !-
2030 3542 2
2031 3543 2  IF ( .YES_CONV )
2032 3544 2  THEN
2033 3545 2      CONV_OK = COB$$$ACC_CONVERT ( .STRING_DEST, .FLAGS,
2034 3546 2      .DEFAULT, .PUT_HERE, .CHARS_READ,
2035 3547 2      .YES_DEFAULT, .YES_SIGN )
2036 3548 2  ELSE
2037 3549 2      BEGIN
2038 3550 2          LOCAL
2039 3551 2          COPY_NUM ;
2040 3552 2
2041 3553 2      !+
2042 3554 2      !- No conversion requested - copy input data to STRING_DEST.
2043 3555 2      !- Use STR$COPY_R because it BLANK fills.
2044 3556 2
2045 3557 2
2046 3558 2      IF .CHARS_READ LSS .STRING_DEST[DSC$W_LENGTH]
2047 3559 2      THEN
2048 3560 2          COPY_NUM = .CHARS_READ
2049 3561 2      ELSE
2050 3562 2          COPY_NUM = .STRING_DEST[DSC$W_LENGTH] ;
2051 3563 2
2052 3564 2      STR$COPY_R ( .STRING_DEST, COPY_NUM,
2053 3565 2          (IF .YES_DEFAULT
2054 3566 2          THEN .DEFAULT [DSC$A_POINTER]
2055 3567 2          ELSE .PUT_HERE [DSC$A_POINTER] )) ;
2056 3568 2
2057 3569 2      CONV_OK = 1 ;
2058 3570 2      ! set CONV_OK to success
2059 3571 2      END;
2060 3572 2
2061 3573 2      !+
2062 3574 2      !- Conversion completed - was it successful ?
2063 3575 2
2064 3576 2      IF .CONV_OK EQL 0
2065 3577 2      THEN
2066 3578 2          !+
2067 3579 2          !- CONVERSION error. Read UNIT parameter to determine what
2068 3580 2          !- to do. There is no Reprompting done with files as input.
2069 3581 2
2070 3582 2          Byte 2 of
2071 3583 2          UNIT
2072 3584 2          Conversion
2073 3585 2          error
2074 3586 2          0
2075 3587 2          COB$_ERRDURACC
2076 3588 2          1 ( at end )
2077 3589 2          COB$_ERRDURACC
2078 3590 2          2 ( on exception )
2079 3591 2          Return
2080 3592 2
2081 3593 2      BEGIN
2082 3594 2          ! Begin conversion error
2083 3595 2          IF ( .FLAGS AND V_COB_RPG ) NEQ 0
2084 3596 2          THEN
2085 3597 2              !+
2086 3598 2              !- VAX RPG - return on a Conversion Error, ring bell
2087 3599 2              !- and clean up first.
2088 3600 2              BEGIN
```



```

2086 3598 4 COB$$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
2087 3599 4 COB$$$RPG_CLEAN_UP ( .FLAGS ) ;
2088 3600 4 RETURN 0 ;
2089 3601 3 END ;
2090 3602 3
2091 3603 3 IF .UNIT [1] EQL 2
2092 3604 3 THEN
2093 3605 3 RETURN 0
2094 3606 3 ELSE
2095 3607 3
2096 3608 3 !+
2097 3609 3 When dealing with a file, it was decided to return a fatal
2098 3610 3 error message rather than REBPROMPT. This lets the user know
2099 3611 3 where the problem in the file is so that the input file can
2100 3612 3 be corrected before running the program again. Otherwise, the
2101 3613 3 the user might run out of data by the end of the program or the
2102 3614 3 reprompting process may lead to further conversion errors.
2103 3615 3 !-
2104 3616 3
2105 3617 3 LIB$$STOP ( COB$_ERRDURACC ) ;
2106 3618 3 END ; ! End conversion error
2107 3619 3
2108 3620 3 !+
2109 3621 3 Fill in optional LENGTH parameter with the number of
2110 3622 3 characters read if no error.
2111 3623 3 !-
2112 3624 3
2113 3625 3 IF .LENGTH NEQ 0
2114 3626 3 THEN
2115 3627 3 .LENGTH = .CHARS_READ ;
2116 3628 3
2117 3629 3 *****
2118 3630 3 ***** CLEAN UP
2119 3631 3 *****
2120 3632 3
2121 3633 3 !+
2122 3634 3 Determine if ADVANCING is requested.
2123 3635 3 If bit 10 = 0 advancing. If bit 10 = 1 no advancing.
2124 3636 3 Set COB$$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
2125 3637 3 statement that advancing/no advancing is required following this
2126 3638 3 ACCEPT statement. Echo carriage return to screen if advancing is
2127 3639 3 called for.
2128 3640 3 !-
2129 3641 3
2130 3642 3 IF (.FLAGS AND V_ADV) NEQ 0
2131 3643 3 THEN
2132 3644 3 COB$$$AB_PREV[0] = ACC_DNA ! No Advancing
2133 3645 3 ELSE
2134 3646 3 COB$$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ; ! Advance via a carriage
2135 3647 3 ! return
2136 3648 3 RETURN 1 ;
2137 3649 3 END ; ! End of COB$$$ACC_SCR_FILE

```



```
07FC 00000 COB$$$ACC_SCR FILE:
5A      0000V  CF 9E 00002  .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9,R10      : 3314
59      00000000G 8F D0 00007  MOVAB      COB$$$RMS_PUT_BYTE, R10
58      00000000G 00 9E 0000E  MOVL       #COB$ ERRDURACC, R9
57      00000000G 00 9E 00015  MOVAB      COB$$$AB_PREV, R8
5E      00000000G 04 C2 0001C  MOVAB      LIB$STOP, R7
55      00000000G 55 7C 0001F  SUBL2     #4, SP
54      00000000G 54 D4 00021  CLRQ      CHARS_READ
50      00000000G 68 9A 00023  CLRL     YES_DEFAULT
02      00000000G 0A 13 00026  MOVZBL   COB$$$AB_PREV, R0
04      00000000G 05 13 00028  BEQL     1$
05      00000000G 50 91 0002B  CMPB     R0, #2
06      00000000G 08 12 0002D  BEQL     1$
07      00000000G 08 12 00030  CMPB     R0, #4
0C      00000000G 01 DD 00032 1$:  BNEQ     2$
01      00000000G 02 FB 00035  PUSHL    FLAGS
02      00000000G 04 AC 9A 0003A 2$:  PUSHL    #1
03      00000000G 18 AC B0 00046  CALLS    #2, COB$$$RMS_PUT_BYTE
04      00000000G 1C AC D0 0004B  MOVZBL   UNIT, R0
05      00000000G 10 8A 00054  MOVL     COB$$$AL_WRITE_RAB[R0], RAB
06      00000000G 52 DD 00058 3$:  MOVW     ACC_SIZE, 32(RAB)
07      00000000G 01 FB 0005A  MOVL     PUT_HERE, R3
08      00000000G 50 D1 00061  MOVL     4(R3), 36(RAB)
09      00000000G 0B 12 00068  BICB2    #16, 7(RAB)
0A      00000000G 52 DD 0006A 4$:  PUSHL    RAB
0B      00000000G 01 FB 0006C  CALLS    #1, SYSS$WAIT
0C      00000000G 03 11 00073  BRB      3$
0D      00000000G 08 A2 E8 00075 5$:  BLBS     8(RAB), 10$
0E      00000000G 06 C 91 00079 4$:  CMPB     (AP), #4
0F      00000000G 05 1F 0007C  BLSSU    5$
10      00000000G 32 12 00081  TSTL     16(AP)
11      00000000G 08 A2 7D 00083 5$:  BNEQ     10$
12      00000000G 44 A2 9F 00087  MOVQ     8(RAB), -(SP)
13      00000000G 01 DD 0008A  PUSHAB   68(RAB)
14      00000000G 08 A2 D1 0008C  PUSHL    #1
15      00000000G 1A 12 00094  CMPL     8(RAB), #98938
16      00000000G 04 13 0009A  BNEQ     8$
17      00000000G 01 05 AC 91 00096  CMPB     UNIT+1, #1
18      00000000G 02 05 AC 91 0009C  BEQL     6$
19      00000000G 03 12 000A0 6$:  CMPB     UNIT+1, #2
20      00000000G 00F9 31 000A2 7$:  BNEQ     7$
21      00000000G 50 8F D0 000A5  MOVL     #COB$_EOFON_ACC, R0
22      00000000G 50 DD 000AC  PUSHL    R0
23      00000000G 02 11 000AE  BRB      9$
24      00000000G 59 DD 000B0 8$:  PUSHL    R9
25      00000000G 67 05 FB 000B2 9$:  CALLS    #5, LIB$STOP
26      00000000G 55 22 A2 3C 000B5 10$: MOVZWL   34(RAB), CHARS_READ
27      00000000G 50 D4 000B9  CLRL     R0
28      00000000G 55 D5 000BB  TSTL     CHARS_READ
29      00000000G 15 12 000BD  BNEQ     12$
30      00000000G 50 D6 000BF  INCL     R0
31      00000000G 0B E1 000C1  BBC      #11, FLAGS, 12$
OE      0C      AC      0B      E1      000C1
```



	03	OC	AC	0A	E1	000C6	BBC	#10, FLAGS, 11\$	: 3486
				00BF	31	000CB	BRW	23\$	:
			68	04	90	000CE	MOV B	#4, COB\$\$\$AB_PREV	: 3490
				00C6	31	000D1	BRW	25\$	: 3491
			36	50	E9	000D4	BLBC	R0, 13\$	: 3503
				10	AC	D5	TSTL	DEFAULT	: 3506
					31	13	BEQL	13\$	:
					54	D5	TSTL	YES_DEFAULT	:
					2D	12	BNEQ	13\$	:
			55	10	BC	3C	MOVZWL	@DEFAULT, CHARS_READ	: 3510
			54		01	D0	MOVL	#1, YES_DEFAULT	: 3511
			22	24	AC	E9	BLBC	YES_PROTECT, 13\$	: 3519
			50	08	AC	D0	MOVL	STRING_DEST, R0	: 3520
			0A	02	A0	91	CMPB	2(R0), #10	:
					18	13	BEQL	13\$	:
			0B	02	A0	91	CMPB	2(R0), #11	: 3521
					12	13	BEQL	13\$	:
18	AC		10	BC	00	ED	CMPZV	#0, #16, @DEFAULT, ACC_SIZE	: 3523
					09	15	BLEQ	13\$	:
				00000000G	8F	DD	PUSHL	#COB\$ INVDEFVAL	: 3530
			67		01	FB	CALLS	#1, LIB\$STOP	:
			1A	20	AC	E9	BLBC	YES_CONV, 14\$	: 3543
				28	AC	DD	PUSHL	YES_SIGN	: 3547
					54	DD	PUSHL	YES_DEFAULT	:
					28	3B	PUSHR	#^MZR3,R5>	: 3546
			7E	0C	AC	7D	MOVQ	FLAGS, -(SP)	: 3545
				08	AC	DD	PUSHL	STRING_DEST	:
			00000000G	00	07	FB	CALLS	#7, COB\$\$\$ACC_CONVERT	:
			56		50	D0	MOVL	R0, CONV_OK	:
					30	11	BRB	19\$	:
55			08	BC	00	ED	CMPZV	#0, #16, @STRING_DEST, CHARS_READ	: 3558
					05	15	BLEQ	15\$	:
			6E		55	D0	MOVL	CHARS_READ, COPY_NUM	: 3560
					04	11	BRB	16\$	:
			6E	08	BC	3C	MOVZWL	@STRING_DEST, COPY_NUM	: 3562
			09		54	E9	BLBC	YES_DEFAULT, 17\$	: 3565
			50	10	AC	D0	MOVL	DEFAULT, R0	: 3566
				04	A0	DD	PUSHL	4(R0)	:
					03	11	BRB	18\$	:
				04	A3	DD	PUSHL	4(R3)	: 3567
				04	AE	9F	PUSHAB	COPY_NUM	: 3564
				08	AC	DD	PUSHL	STRING_DEST	:
			00000000G	00	03	FB	CALLS	#3, STR\$COPY_R	:
			56		01	D0	MOVL	#1, CONV_OK	: 3569
					22	12	BNEQ	21\$	: 3576
			12	OC	AC	0B	BBC	#11, FLAGS, 20\$	: 3591
					0C	AC	PUSHL	FLAGS	: 3598
					02	DD	PUSHL	#2	:
			6A		02	FB	CALLS	#2, COB\$\$\$RMS_PUT_BYTE	: 3599
				0C	AC	DD	PUSHL	FLAGS	:
			0000V	CF	01	FB	CALLS	#1, COB\$\$\$RPG_CLEAN_UP	: 3600
					2A	11	BRB	26\$	: 3603
			02	05	AC	91	CMPB	UNIT+1, #2	:
					24	13	BEQL	26\$	: 3617
					59	DD	PUSHL	R9	:
			67		01	FB	CALLS	#1, LIB\$STOP	: 3625
				14	AC	D5	TSTL	LENGTH	:



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COBSACC\_SCR\_FILE - Screen enhancements for file

6  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 55  
(5)

05	14	BC	04	13	00182	BEQL	22\$	:	3627	
	OC	AC	55	D0	00184	MOVL	CHARS_READ, @LENGTH	:	3642	
		68	0A	E1	00188	BBC	#10, FLAGS, 24\$	:	3644	
			05	90	0018D	MOVB	#5, COB\$\$AB_PREV	:	3646	
			08	11	00190	BRB	25\$	:	3648	
			OC	AC	DD	00192	PUSHL	FLAGS	:	3649
			7E	D4	00195	CLRL	-(SP)	:		
	6A		02	FB	00197	CALLS	#2, COB\$\$RMS_PUT_BYTE	:		
	50		01	D0	0019A	MOVL	#1, R0	:		
				04	0019D	RET		:		
			50	D4	0019E	CLRL	R0	:		
				04	001A0	RET		:		

; Routine Size: 417 bytes, Routine Base: \_COB\$CODE + 09AA



```
2139 3650 1 %SBTTL 'COBS$OPEN IN - Open for INPUT'
2140 3651 1 GLOBAL ROUTINE COBS$OPEN_IN (UNIT, RPG): NOVALUE =
2141 3652 1
2142 3653 1 !++
2143 3654 1 FUNCTIONAL DESCRIPTION:
2144 3655 1
2145 3656 1     Open a file for reading, given its unit number.
2146 3657 1
2147 3658 1 FORMAL PARAMETERS:
2148 3659 1
2149 3660 1     UNIT.rl.v     integer unit number designating the device
2150 3661 1                  from which the string is to be read.
2151 3662 1
2152 3663 1     RPG.rl.v       = 1 if RPG is calling this routine
2153 3664 1                  = 0 if COBOL is calling this routine
2154 3665 1
2155 3666 1 IMPLICIT INPUTS:
2156 3667 1
2157 3668 1     NONE
2158 3669 1
2159 3670 1 IMPLICIT OUTPUTS:
2160 3671 1
2161 3672 1     NONE
2162 3673 1
2163 3674 1 ROUTINE VALUE:
2164 3675 1
2165 3676 1     NONE
2166 3677 1
2167 3678 1 SIDE EFFECTS:
2168 3679 1
2169 3680 1     NONE
2170 3681 1
2171 3682 1 --
2172 3683 1
2173 3684 2 BEGIN
2174 3685 2 LITERAL
2175 3686 2     MAX_BUF = MAX(LNM$C_NAMLENGTH, NAM$C_MAXRSS);
2176 3687 2 LOCAL
2177 3688 2     FAB: $FAB_DECL,
2178 3689 2     NAM: $NAM_DECL,
2179 3690 2     RAB: REF $RAB_DECL,
2180 3691 2     FILE_NAME: BLOCK[8, BYTE], ! Descriptor for the file name
2181 3692 2     TRANSLATE: BLOCK[8, BYTE],
2182 3693 2     P: REF VECTOR[BYTE],
2183 3694 2     RSLBUF: VECTOR[MAX_BUF, BYTE],
2184 3695 2     STATUS;
2185 3696 2
2186 3697 2
2187 3698 2 ! Determine whether the COB$xxx name is defined.
2188 3699 2 ! If so, use it. If not, use the corresponding SY$xxx name.
2189 3700 2
2190 3701 2 TRANSLATE[DSC$B_DTYPE] = DSC$K_DTYPE_T;
2191 3702 2 TRANSLATE[DSC$B_CLASS] = DSC$K_CLASS_S;
2192 3703 2 TRANSLATE[DSC$W_LENGTH] = MAX_BUF;
2193 3704 2 TRANSLATE[DSC$A_POINTER] = RSLBUF;
2194 3705 2
2195 3706 2 !+
```



```
: 2196      3707 2      ! If VAX RPG is calling this routine, bypass COB_TABLE.
: 2197      3708 2      !-
: 2198      3709 2
: 2199      3710 2      IF .RPG
: 2200      3711 2      THEN
: 2201      3712 2          BEGIN                                ! Use the SYS$xxx logical
: 2202      3713 2          P = .SYS_TABLE[.UNIT] + BASE;
: 2203      3714 2          FILE_NAME[DSC$W_LENGTH] = .P[0];
: 2204      3715 2          FILE_NAME[DSC$A_POINTER] = P[1];
: 2205      3716 2          END
: 2206      3717 2      ELSE
: 2207      3718 2          BEGIN                                ! Use the COB$xxx logical
: 2208      3719 2          P = .COB_TABLE[.UNIT] + BASE;
: 2209      3720 2          FILE_NAME[DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 2210      3721 2          FILE_NAME[DSC$B_CLASS] = DSC$K_CLASS_S;
: 2211      3722 2          FILE_NAME[DSC$W_LENGTH] = .P[0];
: 2212      3723 2          FILE_NAME[DSC$A_POINTER] = P[1];
: 2213      3724 2          IF $TRNLOG(LOGNAM = FILE_NAME, RSLBUF = TRANSLATE) NEQ SS$NORMAL
: 2214      3725 2          THEN
: 2215      3726 2              BEGIN                                ! Use the SYS$xxx logical
: 2216      3727 2              P = .SYS_TABLE[.UNIT] + BASE;
: 2217      3728 2              FILE_NAME[DSC$W_LENGTH] = .P[0];
: 2218      3729 2              FILE_NAME[DSC$A_POINTER] = P[1];
: 2219      3730 2              END;
: 2220      3731 2          END ;
: 2221      3732 2
: 2222      P 3733 2      $FAB_INIT(
: 2223      P 3734 2          FAB = FAB,
: 2224      P 3735 2          NAM = NAM,
: 2225      P 3736 2          FAC = <GET,PUT>,
: 2226      P 3737 2          FNA = .FILE_NAME[DSC$A_POINTER],
: 2227      P 3738 2          FNS = .FILE_NAME[DSC$W_LENGTH],
: 2228      3739 2          FOP = SQO);
: 2229      3740 2
: 2230      P 3741 2      $NAM_INIT(
: 2231      P 3742 2          NAM = NAM,
: 2232      P 3743 2          ESA = RSLBUF,
: 2233      P 3744 2          ESS = NAM$C_MAXRSS,
: 2234      P 3745 2          RSA = RSLBUF,
: 2235      3746 2          RSS = NAM$C_MAXRSS);
: 2236      3747 2
: 2237      3748 2      STATUS = $OPEN(FAB = FAB);
: 2238      3749 2      IF (TRANSLATE[DSC$W_LENGTH] = .NAM[NAM$B_RSL]) EQL 0 THEN
: 2239      3750 2      IF (TRANSLATE[DSC$W_LENGTH] = .NAM[NAM$B_ESL]) EQL 0
: 2240      3751 2      THEN
: 2241      3752 2          BEGIN
: 2242      3753 2              TRANSLATE[DSC$W_LENGTH] = .FAB[FAB$B_FNS];
: 2243      3754 2              TRANSLATE[DSC$A_POINTER] = .FAB[FAB$L_FNA];
: 2244      3755 2          END;
: 2245      3756 2
: 2246      3757 2
: 2247      3758 2      IF NOT .STATUS
: 2248      3759 2      THEN
: 2249      3760 2          LIB$STOP(COB$ERRDURACC, 1, TRANSLATE, .FAB[FAB$L_STS], .FAB[FAB$L_STV]);
: 2250      3761 2
: 2251      3762 2
: 2252      3763 2      IF NOT (STATUS = LIB$GET_VM(%REF(RAB$C_BLN + 8 + .NAM[NAM$B_RSL]), RAB))
```



```
! End of COB$$$OPEN_IN
```

.ENTRY	COB\$\$OPEN_IN, Save R2,R3,R4,R5,R6,R7,R8,R9,-;	3651
	R10,R11	:
MOVAB	LIB\$STOP, R11	:
MOVAB	\$RMS_PTR, R10	:
MOVAB	-456(SP), SP	:
MOVL	#17694975, TRANSLATE	3703
MOVAB	RSLBUF, TRANSLATE+4	3704
MOVL	UNIT, R8	3713
ASHL	#2, R8, R7	:
BLBS	RP6, 1\$	3710
MOVAB	BASE, R0	3719

57



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$OPEN\_IN - Open for INPUT

F 6  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 59  
(6)

52		50	F4E3 CF47	9F 00035	PUSHAB	COB TABLE[R7]	
	FF4A	CD	010E 8F	C1 0003A	ADDL3	@(SP)+, R0, P	3720
	FF48	CD	62 8B	80 0003E	MOVW	#270, FILE_NAME+2	3722
	FF4C	CD	01 A2	9B 00045	MOVZBW	(P), FILE_NAME	3723
			7E 7C	9E 0004A	MOVAB	1(R2), FILE_NAME+4	3724
			7E D4	00050	CLRQ	-(SP)	
			7E D4	00052	CLRL	-(SP)	
			FF40 CD	9F 00054	PUSHAB	TRANSLATE	
			7E D4	00058	CLRL	-(SP)	
			FF48 CD	9F 0005A	PUSHAB	FILE_NAME	
00000000G	00	01	06 FB	0005E	CALLS	#6, SYS\$TRNLOG	
			50 D1	00065	CMPL	R0, #1	
			19 13	00068	BEQL	2\$	
		50	F447 CF	9E 0006A	MOVAB	BASE, R0	3727
			F511 CF47	9F 0006F	PUSHAB	SYS TABLE[R7]	
52		50	9E C1	00074	ADDL3	@(SP)+, R0, P	
	FF48	CD	62 9B	00078	MOVZBW	(P), FILE_NAME	3728
	FF4C	CD	01 A2	9E 0007D	MOVAB	1(R2), FILE_NAME+4	3729
0050 8F	00	6E	00 2C	00083	MOVCS	#0, (SP), #0, #80, \$RMS_PTR	3739
			B0 AD	0008A			
	B0 AD		5003 8F	B0 0008C	MOVW	#20483, \$RMS_PTR	
	B4 AD		40 8F	9A 00092	MOVZBL	#64, \$RMS_PTR+4	
	C6 AD		03 90	00097	MOVB	#3, \$RMS_PTR+22	
	CF AD		02 90	0009B	MOVB	#2, \$RMS_PTR+31	
	D8 AD		FF50 CD	9E 0009F	MOVAB	NAM, \$RMS_PTR+40	
	DC AD		FF4C CD	D0 000A5	MOVL	FILE_NAME+4, \$RMS_PTR+44	
0060 8F	E4 AD		FF48 CD	90 000AB	MOVB	FILE_NAME, \$RMS_PTR+52	
		6E	00 2C	000B1	MOVCS	#0, (SP), #0, #96, \$RMS_PTR	3746
			FF50 CD	000B8			
	FF50 CD		6002 8F	B0 000BB	MOVW	#24578, \$RMS_PTR	
	FF52 CD		01 8E	000C2	MNEGB	#1, \$RMS_PTR+2	
	FF54 CD		08 AE	9E 000C7	MOVAB	RSLBUF, \$RMS_PTR+4	
	FF5A CD		01 8E	000CD	MNEGB	#1, \$RMS_PTR+10	
	FF5C CD		08 AE	9E 000D2	MOVAB	RSLBUF, \$RMS_PTR+12	
			B0 AD	9F 000D8	PUSHAB	FAB	3748
00000000G	00		01 FB	000DB	CALLS	#1, SYS\$OPEN	
	52		50 D0	000E2	MOVL	R0, STATUS	
	FF40 CD		FF53 CD	9B 000E5	MOVZBW	NAM+3, TRANSLATE	3749
			15 12	000EC	BNEQ	3\$	
	FF40 CD		FF5B CD	9B 000EE	MOVZBW	NAM+11, TRANSLATE	3750
			0C 12	000F5	BNEQ	3\$	
	FF40 CD		E4 AD	9B 000F7	MOVZBW	FAB+52, TRANSLATE	3753
	FF44 CD		DC AD	D0 000FD	MOVL	FAB+44, TRANSLATE+4	3754
		13	52 E8	00103	BLBS	STATUS, 4\$	3758
		7E	B8 AD	7D 00106	MOVQ	FAB+8, -(SP)	3760
			FF40 CD	9F 0010A	PUSHAB	TRANSLATE	
			01 DD	0010E	PUSHL	#1	
			8F DD	00110	PUSHL	#COB\$ ERRDURACC	
		6B	05 FB	00116	CALLS	#5, LIB\$STOP	
	04 AE		04 AE	9F 00119	PUSHAB	RAB	3763
	04 AE		FF53 CD	9A 0011C	MOVZBL	NAM+3, 4(SP)	
			8F C0	00122	ADDL2	#76, 4(SP)	
			04 AE	9F 0012A	PUSHAB	4(SP)	
00000000G	00		02 FB	0012D	CALLS	#2, LIB\$GET_VM	
	52		50 D0	00134	MOVL	R0, STATUS	
	0D		52 E8	00137	BLBS	STATUS, 5\$	
			52 DD	0013A	PUSHL	STATUS	3765



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$OPEN\_IN - Open for INPUT

G 6  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 60  
(6)

				00000000G	7E	D4	0013C	CLRL	-(SP)	
					8F	DD	0013E	PUSHL	#COB\$ FAIGET_VM	
				6B	03	FB	00144	CALLS	#3, LIB\$STOP-	
				56	04	AE	D0 00147	5\$:	MOVL	RAB, R6
				59	44	A6	9E 0014B	MOVAB	68(R6), R9	3774
				50		59	D0 0014F	MOVL	R9, Q	
				02	A0	010E	8F B0 00152	MOVW	#270, 2(Q)	3775
				60	FF40	CD	B0 00158	MOVW	TRANSLATE, (Q)	3777
				04	A0	4C	A6 9E 0015D	MOVAB	76(R6), 4(Q)	3778
				DD		60	28 00162	MOVC3	(Q), @TRANSLATE+4, 76(R6)	3779
				6E		00	2C 00169	MOVC5	#0, (SP), #0, #36, \$RMS_PTR	3788
						6A	0016E			
				6A	241F	8F	B0 0016F	MOVW	#9247, \$RMS_PTR	
				08	AA	AA	9E 00174	MOVAB	XAB_ITMLST, \$RMS_PTR+8	
				OC	AA	18	B0 00179	MOVW	#24, \$RMS_PTR+12	
0044	3F		00	6E		00	2C 0017D	MOVC5	#0, (SP), #0, #68, (R6)	3793
						66	00184			
				66	4401	8F	B0 00185	MOVW	#17409, (R6)	
				3C	A6	AD	9E 0018A	MOVAB	FAB, 60(R6)	
				40	A6	6A	9E 0018F	MOVAB	XABTRM, 64(R6)	
						56	DD 00193	PUSHL	R6	3795
				00000000G	00	01	FB 00195	CALLS	#1, SYSS\$CONNECT	
					11	50	E8 0019C	BLBS	R0, 6\$	
					7E	08	A6 7D 0019F	MOVQ	8(R6), -(SP)	3797
						59	DD 001A3	PUSHL	R9	
						01	DD 001A5	PUSHL	#1	
						8F	DD 001A7	PUSHL	#COB\$ ERRDURACC	
				6B		05	FB 001AD	CALLS	#5, LIB\$STOP	
				00000000G	0047	9F	001B0	6\$:	COB\$\$AL WRITE_RAB[R7]	3799
				9E	56	D0	001B7	MOVL	R6, @SP+	
				00000000G	0048	B2	AD	MOVW	FAB+2, COB\$\$AW_WRITE_IFI[R8]	3800
						04	001C3	RET		3802

; Routine Size: 452 bytes, Routine Base: \_COB\$CODE + 0B4B



```
2293 3803 1 %SBTTL 'COB$$RMS_GET - Perform an RMS $GET Service'
2294 3804 1 ROUTINE COB$$RMS_GET ( RAB : REF $RAB_DECL,
2295 3805 1     FUNC_VAL,
2296 3806 1     LENGTH,
2297 3807 1     BUFFER
2298 3808 1 ) : NOVALUE =
2299 3809 1
2300 3810 1 ++
2301 3811 1 FUNCTIONAL DESCRIPTION:
2302 3812 1
2303 3813 1
2304 3814 1 FORMAL PARAMETERS:
2305 3815 1
2306 3816 1
2307 3817 1 IMPLICIT INPUTS:
2308 3818 1
2309 3819 1     NONE
2310 3820 1
2311 3821 1 IMPLICIT OUTPUTS:
2312 3822 1
2313 3823 1     NONE
2314 3824 1
2315 3825 1 ROUTINE VALUE:
2316 3826 1 COMPLETION CODES:
2317 3827 1
2318 3828 1     NONE
2319 3829 1
2320 3830 1 SIDE EFFECTS:
2321 3831 1
2322 3832 1 --
2323 3833 1
2324 3834 2 BEGIN
2325 3835 2
2326 3836 2 $ITMLST_INIT (ITMLST = XAB ITMLST, ! Item list for $GET
2327 3837 2     (ITMCOB = TRMS_MODIFIERS,
2328 3838 2     BUFSIZ = 0,
2329 3839 2     BUFADR = .FUNC_VAL),
2330 3840 2     (ITMCOB = TRMS_TERM,
2331 3841 2     BUFSIZ = 20,
2332 3842 2     BUFADR = MASK_VECTOR) ) ;
2333 3843 2
2334 3844 2 RAB [RAB$W_USZ] = .LENGTH ;
2335 3845 2 RAB [RAB$L_UBF] = .BUFFER ;
2336 3846 2 RAB [RAB$V_ETO] = 1 ; ! Extended Terminal $GET
2337 3847 2 RAB [RAB$L_XAB] = XABTRM ;
2338 3848 2 WHILE $GET (RAB = .RAB) EQL RMSS_RSA DO $WAIT (RAB = .RAB) ;
2339 3849 2
2340 3850 2 IF NOT .RAB [RAB$L_STS]
2341 3851 2 THEN
2342 3852 2     +
2343 3853 2     These are special case status that will be handled later.
2344 3854 2     -
2345 3855 2     IF (.RAB [RAB$L_STS] NEQ RMSS_BES AND ! Bad Escape Sequence
2346 3856 2     .RAB [RAB$L_STS] NEQ RMSS_EOF AND ! End Of File
2347 3857 2     .RAB [RAB$L_STS] NEQ RMSS_PES AND ! Partial Escape Seq
2348 3858 2     .RAB [RAB$L_STS] NEQ RMSS_RTG AND ! Record Too Big
2349 3859 2     .RAB [RAB$L_STS] NEQ RMSS_TNS ) ! Terminator Not Seen
```



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$RMS\_GET - Perform an RMS \$GET Service

I 6  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COB\$ACCEPT.B32;2

Page 62  
(7)

```
: 2350      3860  2      THEN
: 2351      3861  2
: 2352      3862  2      LIB$STOP (COB$_ERRDURACC, 1, .RAB + RAB$_BLN, .RAB [RAB$_STS],
: 2353      3863  1      END ;      .RAB [RAB$_STV] ) ;
```

```
000C 00000 COB$$RMS_GET:
53 00000000' EF 9E 00002 .WORD Save R2,R3 : 3804
50 63 9E 00009 MOVAB XAB_ITMLST, R3 : 3842
80 08 AC D0 0000E MOVAB XAB_ITMLST, $$ITMBLKPTR
80 80 D4 0000C CLRL ($$ITMBLKPTR)+
80 00030014 8F D0 00012 MOVL FUNC_VAL, ($$ITMBLKPTR)+
80 1C A3 9E 0001B CLRL ($$ITMBLKPTR)+
80 7C 0001F MOVAB #196628, ($$ITMBLKPTR)+
52 04 AC D0 00021 CLRQ MASK_VECTOR, ($$ITMBLKPTR)+
20 A2 0C AC B0 00025 MOVL RAB, R2 : 3844
24 A2 10 AC D0 0002A MOVW LENGTH, 32(R2)
07 A2 10 88 0002F MOVL BUFFER, 36(R2) : 3845
40 A2 DC A3 9E 00033 BISB2 #16, 7(R2) : 3846
00000000G 00 52 DD 00038 1$: MOVAB XABTRM, 64(R2) : 3847
000182DA 8F 52 DD 00038 R2 : 3848
00000000G 00 01 FB 0003A CALLS #1, SYSSGET
00000000G 00 50 D1 00041 CMPL R0, #99034
00000000G 00 0B 12 00048 BNEQ 2$
00000000G 00 52 DD 0004A PUSHL R2
00000000G 00 01 FB 0004C CALLS #1, SYSSWAIT
00000000G 50 08 A2 D0 00055 2$: BRB 1$ : 3850
000181C0 8F 50 E8 00059 MOVL 8(R2), R0 : 3855
0001827A 8F 50 D1 0005C CMPL R0, 3$
000181C8 8F 3B 13 00063 BEQL R0, #98752 : 3856
000181A8 8F 50 D1 00065 CMPL R0, #98938 : 3857
000181B8 8F 32 13 0006C BEQL R0, #98760 : 3858
00000000G 00 50 D1 00077 CMPL R0, #98728 : 3859
00000000G 00 29 13 00075 BEQL R0, #98744 : 3862
00000000G 00 50 D1 00080 CMPL R0, #98744 : 3861
00000000G 00 17 13 00087 BEQL 3$
00000000G 00 0C A2 DD 00089 PUSHL 12(R2)
00000000G 00 50 DD 0008C PUSHL R0
00000000G 00 44 A2 9F 0008E PUSHAB 68(R2)
00000000G 00 01 DD 00091 PUSHL #1
00000000G 00 8F DD 00093 PUSHL #COB$_ERRDURACC
00000000G 00 05 FB 00099 CALLS #5, LIB$STOP : 3863
00000000G 00 04 000A0 3$: RET
```

; Routine Size: 161 bytes, Routine Base: \_COB\$CODE + 0D0F



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement

COB\$\$RMS\_PUT\_BYTE - Perform an RMS \$PUT Service

J 6  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 63  
(8)

```

2355 3864 1 %SBTTL 'COB$$RMS_PUT_BYTE - Perform an RMS $PUT Service'
2356 3865 1 ROUTINE COB$$RMS_PUT_BYTE ( WHICH, FLAGS ) : NOVALUE =
2357 3866 1
2358 3867 1 ++
2359 3868 1 FUNCTIONAL DESCRIPTION:
2360 3869 1
2361 3870 1 This routine writes a one byte buffer to the terminal. Either a
2362 3871 1 Carriage Return, Linefeed or Ring the Terminal Bell, depending
2363 3872 1 on the value of WHICH.
2364 3873 1
2365 3874 1 FORMAL PARAMETERS:
2366 3875 1
2367 3876 1 WHICH.rl.v if 0, write Linefeed to terminal
2368 3877 1 if 1, write Carriage Return to terminal
2369 3878 1 if 2, ring terminal bell
2370 3879 1
2371 3880 1 FLAGS.rlu.v Screen enhancement flag
2372 3881 1
2373 3882 1 IMPLICIT INPUTS:
2374 3883 1
2375 3884 1 NONE
2376 3885 1
2377 3886 1 IMPLICIT OUTPUTS:
2378 3887 1
2379 3888 1 NONE
2380 3889 1
2381 3890 1 ROUTINE VALUE:
2382 3891 1 COMPLETION CODES:
2383 3892 1
2384 3893 1 NONE
2385 3894 1
2386 3895 1 SIDE EFFECTS:
2387 3896 1
2388 3897 1 NONE
2389 3898 1 --
2390 3899 1
2391 3900 2 BEGIN
2392 3901 2 LOCAL
2393 3902 2 RAB : REF $RAB_DECL,
2394 3903 2 CR_BUF : VECTOR [T,BYTE] ;
2395 3904 2
2396 3905 2 IF .COBSACC_TERM_TYPE NEQ UNKNOWN
2397 3906 2 THEN
2398 3907 2 BEGIN
2399 3908 2
2400 3909 2 SELECTONE .WHICH OF
2401 3910 2 SET
2402 3911 2 [0] : CR_BUF [0] = CR ; ! Carriage Return
2403 3912 2
2404 3913 2 [1] : CR_BUF [0] = LF ; ! Linefeed
2405 3914 2
2406 3915 2 [2] : CR_BUF [0] = BELL ; ! Bell
2407 3916 2 TES ;
2408 3917 2
2409 3918 2 COB$$AB_USPCODE [0] = 0 ;
2410 3919 2 COB$$AB_USPCODE [1] = 0 ;
2411 3920 2
```



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$RMS\_PUT\_BYTE - Perform an RMS \$PUT Service

K 6  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 64  
(8)

```

: 2412      3921 3      IF .COB$$AL_WRITE_RAB [1] EQL 0
: 2413      3922 3      THEN
: 2414      3923 4          BEGIN
: 2415      3924 4              +
: 2416      3925 4              Open SYSS$OUTPUT. Second parameter tells COB$$OPEN_OUT whether
: 2417      3926 4              VAX COBOL (0) or VAX RPG (1) is the caller.
: 2418      3927 4              VMS V4 defines SYSS$INPUT as read only, therefore any $PUTs must
: 2419      3928 4              be made through SYSS$OUTPUT. When a terminal is the input device
: 2420      3929 4              for an ACCEPT it is also the OUTPUT device, and must be OPENed
: 2421      3930 4              for both.
: 2422      3931 4              -
: 2423      3932 4          COB$$OPEN_OUT ( 1,
: 2424      3933 4              IF ( .FLAGS AND V_COB_RPG ) NEQ 0
: 2425      3934 4              THEN 1
: 2426      3935 4              ELSE 0 ) ;
: 2427      3936 3      END ;
: 2428      3937 3      RAB = .COB$$AL_WRITE_RAB [1] ;
: 2429      3938 3      RAB [RAB$L_RBF] = CR_BUF [0] ;
: 2430      3939 3      RAB [RAB$W_RSZ] = 1 ;
: 2431      3940 3      WHILE $PUT (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
: 2432      3941 3
: 2433      3942 3      IF NOT .RAB [RAB$L_STS]
: 2434      3943 3      THEN
: 2435      3944 3          LIB$STOP ( COB$ ERRDURACC, 1, .RAB + RAB$C_BLN,
: 2436      3945 3          .RAB [RAB$L_STS], .RAB [RAB$L_STV] ) ;
: 2437      3946 2      END ;
: 2438      3947 1      END ;
```

! End of COB\$\$RMS\_PUT\_BYTE

.EXTRN SYSS\$PUT

000C 00000 COB\$\$RMS\_PUT\_BYTE:

53	00000000G	00	9E	00002	WORD	Save R2,R3	3865
5E		04	C2	00009	MOVAB	COB\$\$AL_WRITE_RAB+4, R3	
	00000000G	00	D5	0000C	SUBL2	#4, SP	3905
		7D	13	00012	TSTL	COB\$ACC_TERM_TYPE	
50	04	AC	D0	00014	BEQL	9\$	3909
		05	12	00018	MOVL	WHICH, R0	3911
6E		0D	90	0001A	BNEQ	1\$	
		12	11	0001D	MOVB	#13, CR_BUF	
01		50	D1	0001F 1\$:	BRB	3\$	3913
		05	12	00022	CMPL	R0, #1	
6E		0A	90	00024	BNEQ	2\$	
		08	11	00027	MOVB	#10, CR_BUF	
02		50	D1	00029 2\$:	BRB	3\$	3915
		03	12	0002C	CMPL	R0, #2	
6E		07	90	0002E	BNEQ	3\$	
	00000000G	00	B4	00031 3\$:	MOVB	#7, CR_BUF	
		63	D5	00037	CLRW	COB\$\$AB_USPCODE	3918
		14	12	00039	TSTL	COB\$\$AL_WRITE_RAB+4	3921
04	08	AC	0B	E1 0003B	BNEQ	6\$	
		01	DD	00040	BBC	#11, FLAGS, 4\$	3933
		02	11	00042	PUSHL	#1	
		7E	D4	00044 4\$:	BRB	5\$	
		01	DD	00046 5\$:	CLRL	-(SP)	
					PUSHL	#1	3932



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$RMS\_PUT\_BYTE - Perform an RMS \$PUT Service

L 6  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 65  
(8)

00000000G	00	02	FB	00048	CALLS	#2, COB\$\$OPEN_OUT	:	
	52	63	D0	0004F	MOVL	COB\$\$AL_WRITE-RAB+4, RAB	:	3937
28	A2	6E	9E	00052	MOVAB	CR_BUF, -40(RAB)	:	3938
22	A2	01	B0	00056	MOVW	#1, 34(RAB)	:	3939
		52	DD	0005A	PUSHL	RAB	:	3940
00000000G	00	01	FB	0005C	CALLS	#1, SY\$\$PUT	:	
000182DA	8F	50	D1	00063	CMPL	R0, #99034	:	
		0B	12	0006A	BNEQ	8\$	:	
		52	DD	0006C	PUSHL	RAB	:	
00000000G	00	01	FB	0006E	CALLS	#1, SY\$\$WAIT	:	
		E3	11	00075	BRB	7\$	:	
	16	08	A2	E8	BLBS	8(RAB), 9\$	:	3942
	7E	08	A2	7D	MOVQ	8(RAB), -(SP)	:	3945
		44	A2	9F	PUSHAB	68(RAB)	:	3944
		01	DD	00082	PUSHL	#1	:	
		8F	DD	00084	PUSHL	#COB\$ ERRDURACC	:	
00000000G	00	05	FB	0008A	CALLS	#5, LIB\$STOP	:	
		04	00091	9\$:	RET		:	3947

; Routine Size: 146 bytes, Routine Base: \_COB\$CODE + 0DB0



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$RMS\_PUT\_BUFFER - Perform RMS \$PUT Service

M 6  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 66  
(9)

```
2440 3948 1 %SBTTL 'COB$$RMS_PUT_BUFFER - Perform RMS $PUT Service'
2441 3949 1 ROUTINE COB$$RMS_PUT_BUFFER ( BUFFER,
2442 3950 1                                     LENGTH,
2443 3951 1                                     FLAGS ) : NOVALUE =
2444 3952 1
2445 3953 1 ++
2446 3954 1 -FUNCTIONAL DESCRIPTION:
2447 3955 1
2448 3956 1     This routine writes buffer of more than one byte to the terminal.
2449 3957 1
2450 3958 1 FORMAL PARAMETERS:
2451 3959 1
2452 3960 1     BUFFER.rt.r      Holds sequence to write to screen
2453 3961 1
2454 3962 1     LENGHT.rlu.v      Length of BUFFER
2455 3963 1
2456 3964 1     FLAGS.rlu.v       Screen enhancement flag
2457 3965 1
2458 3966 1 IMPLICIT INPUTS:
2459 3967 1
2460 3968 1     NONE
2461 3969 1
2462 3970 1 IMPLICIT OUTPUTS:
2463 3971 1
2464 3972 1     NONE
2465 3973 1
2466 3974 1 ROUTINE VALUE:
2467 3975 1 COMPLETION CODES:
2468 3976 1
2469 3977 1     NONE
2470 3978 1
2471 3979 1 SIDE EFFECTS:
2472 3980 1
2473 3981 1     NONE
2474 3982 1 --
2475 3983 1
2476 3984 2 BEGIN
2477 3985 2 LOCAL
2478 3986 2     RAB      : REF $RAB_DECL ;
2479 3987 2
2480 3988 2 IF .COB$ACC_TERM_TYPE NEQ UNKNOWN
2481 3989 2 THEN
2482 3990 3 BEGIN
2483 3991 3
2484 3992 3     COB$$AB_USPCODE [0] = 0 ;
2485 3993 3     COB$$AB_USPCODE [1] = 0 ;
2486 3994 3
2487 3995 3 IF .COB$$AL_WRITE_RAB [1] EQL 0
2488 3996 3 THEN
2489 3997 4 BEGIN
2490 3998 4
2491 3999 4     + Open SYS$OUTPUT. Second parameter tells COB$$OPEN_OUT whether
2492 4000 4     VAX COBOL (0) or VAX RPG (1) is the caller.
2493 4001 4
2494 4002 4     COB$$OPEN_OUT ( 1,
2495 4003 4     IF ( .FLAGS AND V_COB_RPG ) NEQ 0
2496 4004 4     THEN 1
```



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$RMS\_PUT\_BUFFER - Perform RMS \$PUT Service

N 6  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COB\$ACCEPT.B32;2

Page 67  
(9)

```
: 2497      4005      4
: 2498      4006      3
: 2499      4007      3
: 2500      4008      3
: 2501      4009      3
: 2502      4010      3
: 2503      4011      3
: 2504      4012      3
: 2505      4013      3
: 2506      4014      3
: 2507      4015      3
: 2508      4016      3
: 2509      4017      1

      ELSE 0 ) ;
      END ;
      RAB = .COB$$AL WRITE RAB [1] ;
      RAB [RAB$L_RBF] = .BUFFER ;
      RAB [RAB$W_RSZ] = .LENGTH ;
      WHILE $PUT (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;

      IF NOT .RAB [RAB$L_STS]
      THEN
        LIB$STOP ( COB$ ERRDURACC, 1, .RAB + RAB$C_BLN,
                   .RAB [RAB$L_STS], .RAB [RAB$L_STV] ) ;
      END ;
END ;

! End of COB$$RMS_PUT_BUFFER
```

```
000C 00000 COB$$RMS_PUT_BUFFER:
      53 00000000G 00 9E 00002      .WORD      Save R2,R3      : 3949
      00000000G 00 05 00009      MOVAB      COB$$AL_WRITE_RAB+4, R3
      00000000G 62 13 0000F      TSTL      COB$ACC_TERM_TYPE      : 3988
      00000000G 00 B4 00011      BEQL      6$
      63 D5 00017      CLRW      COB$$AB_USPCODE      : 3992
      14 12 00019      TSTL      COB$$AL_WRITE_RAB+4      : 3995
      0B E1 0001B      BNEQ      3$
      01 DD 00020      BBC      #11, FLAGS, 1$      : 4003
      02 11 00022      PUSHL     #1
      7E D4 00024 1$:      BRB      2$
      01 DD 00026 2$:      CLRL      -(SP)
      02 FB 00028      PUSHL     #1      : 4002
      63 D0 0002F 3$:      CALLS     #2, COB$$OPEN_OUT
      28 A2 04 AC D0 00032      MOVL      COB$$AL_WRITE_RAB+4, RAB      : 4007
      22 A2 08 AC B0 00037      MOVL      BUFFER, -40(RAB)      : 4008
      00000000G 00 52 04 AC D0 00037      MOVW      LENGTH, 34(RAB)      : 4009
      000182DA 8F 52 DD 0003C 4$:      PUSHL     RAB      : 4010
      00000000G 00 01 FB 0003E      CALLS     #1, SYSS$PUT
      50 D1 00045      CMPL      R0, #99034
      0B 12 0004C      BNEQ      5$
      52 DD 0004E      PUSHL     RAB
      00000000G 00 01 FB 00050      CALLS     #1, SYSS$WAIT
      16 08 A2 E3 11 00057      BRB      4$
      7E 08 A2 E8 00059 5$:      BLBS      8(RAB), 6$      : 4012
      44 A2 7D 0005D      MOVQ      8(RAB), -(SP)      : 4015
      01 DD 00064      PUSHAB    68(RAB)      : 4014
      00000000G 00 8F DD 00066      PUSHL     #1
      05 FB 0006C      PUSHL     #COB$ ERRDURACC
      04 00073 6$:      CALLS     #5, LIB$STOP
      RET
```

; Routine Size: 116 bytes, Routine Base: \_COB\$CODE + 0E42



```
2511 4018 1 %SBTTL 'COBSSCONTROL_Z - Handle ^Z'
2512 4019 1 ROUTINE COBSSCONTROL_Z ( UNIT      : VECTOR [2, BYTE],
2513 4020 1      KEY      : REF $STR$DESCRIPTOR
2514 4021 1      ) : NOVALUE =
2515 4022 1
2516 4023 1 **
2517 4024 1 FUNCTIONAL DESCRIPTION:
2518 4025 1     Read UNIT parameter to determine what to do when a Control Z was typed.
2519 4026 1
2520 4027 1 FORMAL PARAMETERS:
2521 4028 1
2522 4029 1     UNIT.rbu.va    Array of two unsigned byte integers.
2523 4030 1                 The first byte is the unit number designating the
2524 4031 1                 device from which the string is to be read.
2525 4032 1                 The second byte indicates whether the routine should
2526 4033 1                 abort or return to the calling program.
2527 4034 1
2528 4035 1     KEY.wt.ds      Destination of the receiving field of the control key.
2529 4036 1
2530 4037 1 IMPLICIT INPUTS:
2531 4038 1
2532 4039 1     NONE
2533 4040 1
2534 4041 1 IMPLICIT OUTPUTS:
2535 4042 1
2536 4043 1     NONE
2537 4044 1
2538 4045 1 ROUTINE VALUE:
2539 4046 1
2540 4047 1
2541 4048 1 SIDE EFFECTS:
2542 4049 1
2543 4050 1     NONE
2544 4051 1 --
2545 4052 2 BEGIN
2546 4053 2     LOCAL
2547 4054 2     TERM_PTR,      ! Points to terminator
2548 4055 2     CZ_PTR ;      ! Needed for CH$MOVE
2549 4056 2
2550 4057 2
2551 4058 2 + CONTROL Z - read UNIT parameter to determine what to do.
2552 4059 2
2553 4060 2     Byte 2 of          Ctrl z
2554 4061 2     UNIT
2555 4062 2
2556 4063 2     0                  abort
2557 4064 2     1 ( at end )      return
2558 4065 2     2 ( on exception ) return
2559 4066 2
2560 4067 2
2561 4068 2 IF .UNIT [1] EQL 0
2562 4069 2 THEN
2563 4070 2     LIB$STOP ( COB$_EOFON_ACC )      ! Abort
2564 4071 2 ELSE
2565 4072 2     BEGIN
2566 4073 2     IF .KEY NEQ 0
2567 4074 2     THEN
```



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$CONTROL\_Z - Handle ^Z

C-7  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 69  
(10)

```
: 2568      4075  4      BEGIN
: 2569      4076  4      |*
: 2570      4077  4      |  Pass CONTROL Z back to user program via KEY,
: 2571      4078  4      |  if requested
: 2572      4079  4      |
: 2573      4080  4      | CZ_PTR = CZ ;                ! CZ is literal %X'1A'
: 2574      4081  4      | TERM_PTR = CZ_PTR ;
: 2575      4082  4      | CH$MOVE ( 1, TERM_PTR, .KEY [DSC$A_POINTER] ) ;
: 2576      4083  3      | END ;
: 2577      4084  2      |
: 2578      4085  1      |
                        END ;
                        ! End routine COB$$CONTROL_Z
```

```
0000 00000 COB$$CONTROL_Z:
                        .WORD  Save nothing
                        SUBL2  #4, SP
                        TSTB   UNIT+1
                        BNEQ   1$
00000000G 00 00000000G 8F DD 0000A  PUSHL  #COB$ EOFON ACC
                        01 FB 00010  CALLS  #1, LIB$STOP
                        04 00017  RET
                        50 08 AC D0 00018 1$:  MOVL  KEY, R0
                        0A 13 0001C  BEQL  2$
                        6E 1A D0 0001E  MOVL  #26, CZ_PTR
                        51 6E 9E 00021  MOVAB  CZ_PTR, TERM_PTR
                        04 B0 61 90 00024  MOVB  (TERM_PTR), #4(R0)
                        04 00028 2$:  RET
```

; Routine Size: 41 bytes, Routine Base: \_COB\$CODE + 0EB6



```
: 2580 4086 1 %SBTTL 'COB$$PARTIAL_SEQ - Partial Escape Sequence'
: 2581 4087 1 ROUTINE COB$$PARTIAL_SEQ ( PARAMETERS : REF VECTOR,
: 2582 4088 1                                     UNIT      : VECTOR [2,BYTE] ) : NOVALUE =
: 2583 4089 1
: 2584 4090 1 ++
: 2585 4091 1 FUNCTIONAL DESCRIPTION:
: 2586 4092 1 The entire Escape sequence did not fit in the initial $GET's buffer.
: 2587 4093 1 Perform 1 character Reads until the full sequence is in PUT_HERE or
: 2588 4094 1 NEXT_CHAR.
: 2589 4095 1
: 2590 4096 1 FORMAL PARAMETERS:
: 2591 4097 1
: 2592 4098 1 PARAMETERS.mlu.ra Contains data for this routine.
: 2593 4099 1
: 2594 4100 1 UNIT.rbu.va Array of two unsigned byte integers.
: 2595 4101 1 The first byte is the unit number designating the
: 2596 4102 1 device from which the string is to be read.
: 2597 4103 1 The second byte indicates whether the routine should
: 2598 4104 1 abort or return to the calling program.
: 2599 4105 1
: 2600 4106 1 IMPLICIT INPUTS:
: 2601 4107 1
: 2602 4108 1 NONE
: 2603 4109 1
: 2604 4110 1 IMPLICIT OUTPUTS:
: 2605 4111 1
: 2606 4112 1 NONE
: 2607 4113 1
: 2608 4114 1 ROUTINE VALUE:
: 2609 4115 1
: 2610 4116 1
: 2611 4117 1 SIDE EFFECTS:
: 2612 4118 1
: 2613 4119 1 NONE
: 2614 4120 1
: 2615 4121 1 --
: 2616 4122 2 BEGIN
: 2617 4123 2
: 2618 4124 2 LOCAL
: 2619 4125 2 RAB : REF $RAB_DECL,
: 2620 4126 2 FUNC_VAL_2,
: 2621 4127 2
: 2622 4128 2 TERM_CHAR : BYTE,
: 2623 4129 2 PH : REF VECTOR [1100,BYTE],
: 2624 4130 2 PH_PTR,
: 2625 4131 2 NC_PTR,
: 2626 4132 2 END_OF_TERM : INITIAL (0) ;
: 2627 4133 2
: 2628 4134 2 !+
: 2629 4135 2 ! Bind PARAMETERS to other names.
: 2630 4136 2 !-
: 2631 4137 2
: 2632 4138 2 $BIND_PARAMETERS ;
: 2633 4139 2 PH = .PUT_HERE [D$C$A_POINTER] ;
: 2634 4140 2
: 2635 4141 2 !+
: 2636 4142 2 ! PH_PTR and NC_PTR point to next free space in buffer
```

```
! QIO Function Modifiers for
! item list of RMS $GET.
! $GET input buffer.
! Address of PUT_HERE
! Pointer to PUT_HERE.
! Pointer to NEXT_CHAR.
! =1 whole Seq in buffer.
```



```
2637 4143 2 ! PUT_HERE or NEXT_CHAR.
2638 4144 2 !-
2639 4145 2
2640 4146 2 PH_PTR = .CHARS_READ + .TERM_SIZE ;
2641 4147 2 NC_PTR = 1 ;
2642 4148 2
2643 4149 2 !+
2644 4150 2 Read one character at a time until the entire escape sequence has
2645 4151 2 been read.
2646 4152 2 !-
2647 4153 2
2648 4154 2 WHILE .END_OF_TERM EQL 0 DO
2649 4155 2 BEGIN ! Begin loop
2650 4156 2
2651 4157 2 FUNC_VAL_2 = TRMSM_TM_ESCAPE + TRMSM_TM_NOFILTR + TRMSM_TM_TRMNOECHO
2652 4158 2 + TRMSM_TM_NOECHO ;
2653 4159 2
2654 4160 2 RAB = .COBSSAL_WRITE_RAB [ .UNIT[0] ] ;
2655 4161 2 COBSSRMS_GET ( .RAB, .FUNC_VAL_2, 1, TERM_CHAR ) ;
2656 4162 2
2657 4163 2 !+
2658 4164 2 Deposit sequence character in appropriate buffer.
2659 4165 2 !-
2660 4166 2
2661 4167 2 IF .TERM_IN_NEXT EQL 0
2662 4168 2 THEN
2663 4169 2 BEGIN
2664 4170 2 !+
2665 4171 2 This is a workaround for an RMS bug that did not
2666 4172 2 make it into the final code freeze for V4.0.
2667 4173 2 The next three lines can be pulled when the RMS fix
2668 4174 2 is made. (see NEXT_CHAR below)
2669 4175 2 !-
2670 4176 2 IF .TERM_SIZE EQL 1 ! Put first character
2671 4177 2 THEN ! of terminator seq
2672 4178 2 PH [ .PH_PTR - 1 ] = %X'1B' ; ! into PUT_HERE
2673 4179 2
2674 4180 2 PH [ .PH_PTR ] = .TERM_CHAR ; ! Put character just
2675 4181 2 PH_PTR = .PH_PTR + 1 ; ! read in PUT_HERE
2676 4182 2 TERM_IN_NEXT = 0 ;
2677 4183 2 END
2678 4184 2 ELSE
2679 4185 2 BEGIN
2680 4186 2 !+
2681 4187 2 This is a workaround for an RMS bug that did not
2682 4188 2 make it into the final code freeze for V4.0.
2683 4189 2 The next three lines can be pulled when the RMS fix
2684 4190 2 is made.
2685 4191 2 !-
2686 4192 2 IF .TERM_SIZE EQL 1 ! Put first character
2687 4193 2 THEN ! of terminator seq
2688 4194 2 NEXT_CHAR [0] = %X'1B' ; ! into NEXT_CHAR.
2689 4195 2
2690 4196 2 NEXT_CHAR [ .NC_PTR ] = .TERM_CHAR ; ! Put character just
2691 4197 2 NC_PTR = .NC_PTR + 1 ; ! read in NEXT_CHAR
2692 4198 2 END ;
2693 4199 2
```



```
: 2694      4200  3      TERM_SIZE = .TERM_SIZE + 1 ;      ! Total Terminator size
: 2695      4201  3
: 2696      4202  3
: 2697      4203  3
: 2698      4204  3
: 2699      4205  3
: 2700      4206  3
: 2701      4207  3
: 2702      4208  3
: 2703      4209  3
: 2704      4210  4
: 2705      4211  4
: 2706      4212  4
: 2707      4213  4
: 2708      4214  3
: 2709      4215  4
: 2710      4216  4
: 2711      4217  4
: 2712      4218  4
: 2713      4219  4
: 2714      4220  4
: 2715      4221  4
: 2716      4222  4
: 2717      4223  4
: 2718      4224  3
: 2719      4225  3
: 2720      4226  2
: 2721      4227  1

      Ugly - but it's the only way to check for the end of an
      escape sequence. All known KEY escape sequences end in
      one of these characters and none of these characters fall
      in the middle of an escape sequence. This will have to be
      updated if new escape sequences surface.

      IF ((.TERM_CHAR GEQ %C'A' AND .TERM_CHAR LEQ %C'M') OR
          (.TERM_CHAR GEQ %C'P' AND .TERM_CHAR LEQ %C'S') OR
          (.TERM_CHAR GEQ %C'L' AND .TERM_CHAR LEQ %C'y') OR
          (.TERM_CHAR EQL %X'7E'))
      THEN
      BEGIN
      END_OF_TERM = 1 ;      ! Signal completion
      Have to get rid of a possible status RMS$ TNS, Terminator
      Not Seen. Assume success if we have reached this point.
      It is not advisable to overwrite data in the RAB but there
      is not way to avoid it in this case.
      RAB [RAB$L_STS] = RMS$_SUC ;
      END ;

      END ;      ! End Loop
      END ;      ! End COB$$PARTIAL_SEQ
```

```
07FC 00000 COB$$PARTIAL_SEQ:
      SE      04 C2 00002      .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9,R10      : 4087
      52      04 5A D4 00005      SUBL2      #4, SP
      57      24 AC D0 00007      CLRL      END OF TERM
      53      04 A2 9E 0000B      MOVL      PARAMETERS, R2
      58      1C A2 D0 0000F      MOVAB      36(R2), R7
      56      08 AC 9A 0001B      MOVL      4(R2), PH
      54      01 D0 00018      ADDL3      (R7), 28(R2), PH_PTR
      59      5240 8F 3C 00023      MOVL      #1, NC_PTR
      55 00000000G0044 D0 00028      MOVZBL   UNIT, R4
      5E      DD 00030      TSTL      END_OF_TERM
      01      DD 00032      BNEQ      10$
      FDF3 CF 0220 8F BB 00034      MOVZWL   #21056, FUNC VAL 2
      50      30 6E 9A 0003D      MOVL      COB$$AL_WRITE_RAB[R4], RAB
      01      67 D1 00045      PUSHL      SP
      FF A843 1B 90 0004A      PUSHL      #1
      CMPL      (R7), #1
      BNEQ      2$
      MOV      #27, -1(PH_PTR)[PH]
      4180
      4122
      4132
      4139
      4146
      4147
      4160
      4154
      4158
      4160
      4161
      4176
      4178
```



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COBSPARTIAL\_SEQ - Partial Escape Sequence

6 7  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 73  
(11)

8843		50	90	0004F	2\$:	MOVB	R0, (PH_PTR)+[PH]	: 4180
	30	A2	D4	00053		CLRL	48(R2)	: 4182
		10	11	00056		BRB	5\$	: 4167
01		67	D1	00058	3\$:	CMPL	(R7), #1	: 4192
		04	12	0005B		BNEQ	4\$	: 4194
0C A2		1B	90	0005D		MOVB	#27, 12(R2)	: 4196
0C A246		50	90	00061	4\$:	MOVB	R0, 12(R2)[NC_PTR]	: 4197
		56	D6	00066		INCL	NC_PTR	: 4200
		67	D6	00068	5\$:	INCL	(R7)	: 4210
41 8F		50	91	0006A		CMPB	R0, #65	: 4211
		06	1F	0006E		BLSSU	6\$	: 4212
4D 8F		50	91	00070		CMPB	R0, #77	: 4213
		1E	1B	00074		BLEQU	9\$	: 4214
50 8F		50	91	00076	6\$:	CMPB	R0, #80	: 4215
		06	1F	0007A		BLSSU	7\$	: 4216
53 8F		50	91	0007C		CMPB	R0, #83	: 4217
		12	1B	00080		BLEQU	9\$	: 4218
6C 8F		50	91	00082	7\$:	CMPB	R0, #108	: 4219
		06	1F	00086		BLSSU	8\$	: 4220
79 8F		50	91	00088		CMPB	R0, #121	: 4221
		06	1B	0008C		BLEQU	9\$	: 4222
7E 8F		50	91	0008E	8\$:	CMPB	R0, #126	: 4223
		8B	12	00092		BNEQ	1\$	: 4224
	5A	01	D0	00094	9\$:	MOVL	#1, END_OF_TERM	: 4225
0B A5 00010001		8F	D0	00097		MOVL	#65537, -8(RAB)	: 4226
		FF7D	31	0009F		BRW	1\$	: 4227
		04	000A2	10\$:	RET			: 4228

; Routine Size: 163 bytes, Routine Base: \_COB\$CODE + 0EDF



```

: 2723 4228 1 %SBTTL 'COBSSDELETE_KEY - Delete Key processing'
: 2724 4229 1 ROUTINE COBSSDELETE_KEY ( PARAMETERS : REF VECTOR,
: 2725 4230 1 UNIT : VECTOR [2,BYTE],
: 2726 4231 1 FLAGS ) : NOVALUE =
: 2727 4232 1
: 2728 4233 1 ++
: 2729 4234 1 FUNCTIONAL DESCRIPTION:
: 2730 4235 1 Delete Key processing.
: 2731 4236 1
: 2732 4237 1 FORMAL PARAMETERS:
: 2733 4238 1
: 2734 4239 1 PARAMETERS.mlu.ra Contains data for this routine.
: 2735 4240 1
: 2736 4241 1 UNIT.rbu.va Array of two unsigned byte integers.
: 2737 4242 1 The first byte is the unit number designating the
: 2738 4243 1 device from which the string is to be read.
: 2739 4244 1 The second byte indicates whether the routine should
: 2740 4245 1 abort or return to the calling program.
: 2741 4246 1
: 2742 4247 1 FLAGS.rlu.v Screen enhancement flag.
: 2743 4248 1
: 2744 4249 1 IMPLICIT INPUTS:
: 2745 4250 1
: 2746 4251 1 NONE
: 2747 4252 1
: 2748 4253 1 IMPLICIT OUTPUTS:
: 2749 4254 1
: 2750 4255 1 NONE
: 2751 4256 1
: 2752 4257 1 ROUTINE VALUE:
: 2753 4258 1
: 2754 4259 1
: 2755 4260 1 SIDE EFFECTS:
: 2756 4261 1
: 2757 4262 1 NONE
: 2758 4263 1
: 2759 4264 1 --
: 2760 4265 2 BEGIN
: 2761 4266 2
: 2762 4267 2 LOCAL
: 2763 4268 2 RAB : REF $RAB DECL,
: 2764 4269 2 DELETE_BUF : VECTOR [3, BYTE],
: 2765 4270 2 CHARS_OK : INITIAL (0),
: 2766 4271 2 REST_LEN,
: 2767 4272 2 REST_PTR ;
: 2768 4273 2
: 2769 4274 2
: 2770 4275 2 +
: 2771 4276 2 Delete Key processing. Delete one character at a time. Delete previous
: 2772 4277 2 character typed by backspacing, writing a space to delete (overwrite)
: 2773 4278 2 character, and backspacing again to put cursor in position to continue
: 2774 4279 2 ACCEPTing data.
: 2775 4280 2 Hitting the DELETE KEY terminated the previous $GET, therefore perform
: 2776 4281 2 another $GET to read the rest of the data expected.
: 2777 4282 2
: 2778 4283 2 Bind PARAMETERS to other names.
: 2779 4284 2 -
```

```

! Characters not deleted
! Length yet to be ACCEPTed
! after DELETE KEY was hit
! Where to put rest of
! chars on next $GET
```



```
2780 4285 2
2781 4286
2782 4287
2783 4288
2784 4289
2785 4290
2786 4291
2787 4292
2788 4293
2789 4294
2790 4295
2791 4296
2792 4297
2793 4298
2794 4299
2795 4300
2796 4301
2797 4302
2798 4303
2799 4304
2800 4305
2801 4306
2802 4307
2803 4308
2804 4309
2805 4310
2806 4311
2807 4312
2808 4313
2809 4314
2810 4315
2811 4316
2812 4317
2813 4318
2814 4319
2815 4320
2816 4321
2817 4322
2818 4323
2819 4324
2820 4325
2821 4326
2822 4327
2823 4328
2824 4329
2825 4330
2826 4331
2827 4332
2828 4333
2829 4334
2830 4335
2831 4336
2832 4337
2833 4338
2834 4339
2835 4340
2836 4341

$BIND_PARAMETERS ;
IF ( .FLAGS AND V_NO_ECHO ) NEQ 0
THEN
    +
    If characters were not echoed to terminal there is no need
    to move cursor.
    -
    BEGIN
    DELETE_BUF [0] = NULL ;           ! null
    DELETE_BUF [1] = NULL ;
    DELETE_BUF [2] = NULL ;
    END
ELSE
    BEGIN
    DELETE_BUF [0] = BS ;             ! Backspace
    DELETE_BUF [1] = BLANK ;         ! Space
    DELETE_BUF [2] = BS ;             ! Backspace
    END ;
    WHILE .TERM_LOC EQL DEL_KEY DO
        BEGIN                       ! Begin Delete Loop
            CHARS_READ = (.CHARS_READ - 1) ; ! Decr # of valid input chars
            CHARS_OK = .CHARS_READ ; ! Save for left border check
            IF .CHARS_READ LSS 0
            THEN CHARS_READ = 0 ;
            +
            Calculations for $GET
            -
            REST_LEN = .ACC_SIZE - .CHARS_READ ;
            REST_PTR = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
            +
            Check for too many deletes, do not delete more characters
            then were input. Ring bell to signal attempt to go beyond
            left border.
            -
            IF .CHARS_OK LSS 0
            THEN
                BEGIN
                COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
                END
            ELSE
                +
                $PUT to Delete one character.
                -
                COB$$RMS_PUT_BUFFER ( DELETE_BUF [0], 3, .FLAGS ) ;
            +
            Continue to Read input
            -
            RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
```



```
2837 4342 3 COB$$RMS_GET ( .RAB, .FUNC_VAL, .REST_LEN, .REST_PTR ) ;
2838 4343
2839 4344
2840 4345
2841 4346
2842 4347
2843 4348
2844 4349
2845 4350
2846 4351
2847 4352
2848 4353
2849 4354
2850 4355
2851 4356
2852 4357
2853 4358
2854 4359
2855 4360
2856 4361
2857 4362
2858 4363
2859 4364
2860 4365
2861 4366
2862 4367
2863 4368
2864 4369
2865 4370
2866 4371
2867 4372
2868 4373
2869 4374 1

Reset CHARS_READ - Update # of input chars read.
Reset TERM_SIZE and TERM_LOC - New terminator ( Note: this could
be the DELETE KEY again)

CHARS_READ = .CHARS_READ + .RAB [RAB$W_RSZ] ;
TERM_SIZE = .RAB [COB$$B_STV2_LEN] ;
TERM_LOC = .RAB [COB$$B_STV0_TERM] ; ! Terminator location.

Check for partial sequence error

IF .RAB [RAB$L_STS] EQL RMSS_PES
THEN
COB$$PARTIAL_SEQ ( .PARAMETERS, .UNIT ) ;
END ; ! End Delete Loop

Did the latest $GET come across a terminator ?
If so, set flag used by COB$$ILLEGAL_TERM to signal that the terminator
was encountered in this routine.

IF .TERM_SIZE NEQ 0 OR .RAB [RAB$L_STS] EQL RMSS_EOF
THEN
TERM_FROM_DEL = 1 ;
END ; ! End COB$$DELETE_KEY
```

```
01FC 00000 COB$$DELETE KEY:
07 0C 5E 04 04 C2 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8 : 4229
53 04 58 D4 00005 SUBL2 #4, SP : 4265
55 1C AC D0 00007 CLRL CHARS_OK : 4273
AC 09 A3 9E 0000B MOVL PARAMETERS, R3
6E 02 09 E1 0000F BBC #9, FLAGS, 1$ : 4288
AE 09 8F B4 00014 CLRW DELETE_BUF : 4295
AE 09 94 00016 CLRB DELETE_BUF+2 : 4297
02 09 11 00019 BRB 2$ : 4288
6E 2008 8F B0 0001B 1$: MOVW #8200, DELETE_BUF : 4301
02 AE 08 90 00020 MOVB #8, DELETE_BUF+2 : 4303
54 08 AC 9A 00024 2$: MOVZBL UNIT, R4 : 4341
0000007F 8F 28 A3 D1 00028 3$: CMPL 40(R3), #127 : 4306
6F 12 00030 BNEQ 7$ : 4309
65 D7 00032 DECL (R5) : 4310
65 D0 00034 MOVL (R5), CHARS_OK : 4312
02 18 00037 BGEQ 4$ : 4313
65 D4 00039 CLRL (R5)
```



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$DELETE\_KEY - Delete Key processing

K 7  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 77  
(12)

		57	18	A3	3C	0003B	4\$:	MOVZWL	24(R3), REST_LEN	:	4318
		57		65	C2	0003F		SUBL2	(R5), REST_LEN	:	
56	04	A3		65	C1	00042		ADDL3	(R5), 4(R3), REST_PTR	:	4319
				58	D5	00047		TSTL	CHARS_OK	:	4326
				0C	18	00049		BGEQ	5\$	:	
			0C	AC	DD	0004B		PUSHL	FLAGS	:	4329
				02	DD	0004E		PUSHL	#2	:	
	FDD9	CF		02	FB	00050		CALLS	#2, COB\$\$RMS_PUT_BYTE	:	
				0D	11	00055		BRB	6\$	:	4326
			0C	AC	DD	00057	5\$:	PUSHL	FLAGS	:	4335
				03	DD	0005A		PUSHL	#3	:	
			08	AE	9F	0005C		PUSHAB	DELETE BUF	:	
	FE5C	CF		03	FB	0005F		CALLS	#3, COB\$\$RMS_PUT_BUFFER	:	
		52	00000000G00	44	D0	00064	6\$:	MOVL	COB\$\$AL_WRITE_RAB[R4], RAB	:	4341
				56	DD	0006C		PUSHL	REST_PTR	:	4342
				57	DD	0006E		PUSHL	REST_LEN	:	
			20	A3	DD	00070		PUSHL	32(R3)	:	
				52	DD	00073		PUSHL	RAB	:	
	FD13	CF		04	FB	00075		CALLS	#4, COB\$\$RMS_GET	:	
		50		22	A2	3C	0007A	MOVZWL	34(RAB), R0	:	4350
		65			50	C0	0007E	ADDL2	R0, (R5)	:	
	24	A3		0E	A2	9A	00081	MOVZBL	14(RAB), 36(R3)	:	4351
	28	A3		0C	A2	9A	00086	MOVZBL	12(RAB), 40(R3)	:	4352
000181C8		8F		08	A2	D1	0008B	CMPL	8(RAB), #98760	:	4358
				93	12	00093		BNEQ	3\$	:	
			08	AC	DD	00095		PUSHL	UNIT	:	4360
				53	DD	00098		PUSHL	R3	:	
	FEBE	CF		02	FB	0009A		CALLS	#2, COB\$\$PARTIAL_SEQ	:	
				87	11	0009F		BRB	3\$	:	4306
			24	A3	D5	000A1	7\$:	TSTL	36(R3)	:	4370
				0A	12	000A4		BNEQ	8\$	:	
0001827A		8F		08	A2	D1	000A6	CMPL	8(RAB), #98938	:	
				04	12	000AE		BNEQ	9\$	:	
	34	A3		01	D0	000B0	8\$:	MOVL	#1, 52(R3)	:	4372
				04	000B4	9\$:		RET		:	4374

; Routine Size: 181 bytes, Routine Base: \_COB\$CODE + 0F82



```
2871 4375 1 %SBTTL 'COB$$ILLEGAL_TERM - Illegal Terminator'
2872 4376 1 ROUTINE COB$$ILLEGAL_TERM ( PARAMETERS : REF VECTOR,
2873 4377 1 UNIT : VECTOR [2,BYTE],
2874 4378 1 FLAGS,
2875 4379 1 KEY : REF $STR$DESCRIPTOR ) : NOVALUE =
2876 4380 1
2877 4381 1 ++
2878 4382 1 FUNCTIONAL DESCRIPTION:
2879 4383 1 Terminator from previous $GET was illegal - ring terminal bell to
2880 4384 1 signal this. Perform another $GET of length 1 to look for another
2881 4385 1 terminator. Verify this new terminator.
2882 4386 1
2883 4387 1 FORMAL PARAMETERS:
2884 4388 1
2885 4389 1 PARAMETERS.mlu.ra Contains data for this routine.
2886 4390 1
2887 4391 1 UNIT.rbu.va Array of two unsigned byte integers.
2888 4392 1 The first byte is the unit number designating the
2889 4393 1 device from which the string is to be read.
2890 4394 1 The second byte indicates whether the routine should
2891 4395 1 abort or return to the calling program.
2892 4396 1
2893 4397 1 FLAGS.rlu.v Screen enhancement flag.
2894 4398 1
2895 4399 1 KEY.wt.ds Destination of the receiving field of the control key.
2896 4400 1
2897 4401 1 IMPLICIT INPUTS:
2898 4402 1
2899 4403 1 NONE
2900 4404 1
2901 4405 1 IMPLICIT OUTPUTS:
2902 4406 1
2903 4407 1 NONE
2904 4408 1
2905 4409 1 ROUTINE VALUE:
2906 4410 1
2907 4411 1
2908 4412 1 SIDE EFFECTS:
2909 4413 1
2910 4414 1 NONE
2911 4415 1
2912 4416 1 --
2913 4417 2 BEGIN
2914 4418 2 LOCAL
2915 4419 2 RAB : REF $RAB_DECL,
2916 4420 2 FUNC_VAL_2,
2917 4421 2 NO_BELL : INITIAL (0),
2918 4422 2 LOOK_FOR_TERM : INITIAL (0),
2919 4423 2
2920 4424 2 REST_LEN,
2921 4425 2 REST_PTR ;
2922 4426 2
2923 4427 2 ++
2924 4428 2 Bind PARAMETERS to other names.
2925 4429 2
2926 4430 2
2927 4431 2 $BIND_PARAMETERS ;
```

```
! QIO Function Modifiers
! =0 ring bell, =1 don't
! =1 buffer full, $GET
! only for a terminator
! Length yet to be input
! Where to put rest of
! input data
```



```
2928 4432 2
2929 4433 2
2930 4434 2
2931 4435 2
2932 4436 2
2933 4437 2
2934 4438 2
2935 4439 2
2936 4440 2
2937 4441 2
2938 4442 2
2939 4443 2
2940 4444 2
2941 4445 2
2942 4446 2
2943 4447 2
2944 4448 2
2945 4449 2
2946 4450 2
2947 4451 2
2948 4452 2
2949 4453 2
2950 4454 2
2951 4455 4
2952 4456 4
2953 4457 4
2954 4458 3
2955 4459 3
2956 4460 3
2957 4461 3
2958 4462 3
2959 4463 3
2960 4464 3
2961 4465 3
2962 4466 4
2963 4467 4
2964 4468 4
2965 4469 4
2966 4470 5
2967 4471 5
2968 4472 5
2969 4473 5
2970 4474 5
2971 4475 5
2972 4476 5
2973 4477 5
2974 4478 5
2975 4479 5
2976 4480 5
2977 4481 5
2978 4482 5
2979 4483 5
2980 4484 5
2981 4485 5
2982 4486 5
2983 4487 5
2984 4488 5

Note : If COB$$DELETE_KEY was called before this routine some
special handling is necessary.
It is possible a previous call to COB$$DELETE_KEY would have
filled the input buffer without coming across a terminator.
When the input buffer is full - look for terminator only.

It is also possible that COB$$DELETE_KEY came across a terminator,
therefore it is only necessary to verify the terminator not
perform another $GET. This is flagged by TERM_FROM_DEL = 1.

WHILE .LEGAL EQL 0 DO
  BEGIN
    IF .NO_BELL EQL 0 AND .TERM_FROM_DEL EQL 0
    THEN
      +
      Ring bell to signal illegal terminator.
      Don't ring bell if processing the Delete key, or if the
      terminator has come from COB$$DELETE_KEY (wait for
      $VERIFY_TERMINATOR to check terminator).
      -
      BEGIN
        COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
      END
    ELSE
      NO_BELL = 0 ;
      +
      Is there still data yet to be input ?
      -
      IF .TERM_FROM_DEL EQL 0
      THEN
        BEGIN
          IF .ACC_SIZE GTR .CHARS_READ
          THEN
            BEGIN
              +
              Calculations for $GET.
              -
              REST_LEN = .ACC_SIZE - .CHARS_READ ;
              REST_PTR = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
              +
              NEVER do a Read of 0 length, this causes an infinite loop
              of bell ringing.
              -
              IF .REST_LEN EQL 0 THEN REST_LEN = 1 ;
              RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
              COB$$RMS_GET ( .RAB, .FUNC_VAL, .REST_LEN, .REST_PTR ) ;
              +
              Update CHARS_READ, TERM_SIZE and TERM_LOC.
```



```
2985 4489 5      !-
2986 4490 5
2987 4491 5      CHARS_READ = .CHARS_READ + .RAB [RAB$W_RSZ] ;
2988 4492 5      TERM_SIZE = .RAB [COB$$B_STV2_LEN] ;
2989 4493 5      TERM_LOC = .RAB [COB$$B_STV0_TERM] ;
2990 4494 5
2991 4495 4      END ;
2992 4496 4
2993 4497 4
2994 4498 4      !+
2995 4499 4      $GET buffer filled but no terminator seen - TERM_SIZE = 0
2996 4500 4      Do 1 character reads until you hit a terminator that
2997 4501 4      you can then attempt to verify.
2998 4502 4      Also trap an End of File ^Z here and do not perform
2999 4503 4      another $GET, $VERIFY_TERMINATOR will take care of the ^Z.
3000 4504 4      .LOOK_FOR_TERM EQL 1 case -> came into this routine with
3001 4505 4      $GET buffer filled but illegal terminator, therefore
3002 4506 4      we are looking only for a terminator.
3003 4507 4      !-
3004 4508 4      IF .ACC_SIZE EQL .CHARS_READ
3005 4509 4      THEN LOOK_FOR_TERM = 1 ;
3006 4510 5      WHILE ( .TERM_SIZE EQL 0 AND .RAB [RAB$L_STS] NEQ RMSS_EOF )
3007 4511 4      OR ( .LOOK_FOR_TERM EQL 1 ) DO
3008 4512 4
3009 4513 5      BEGIN                                ! Begin 1 char $GET
3010 4514 5      REST_PTR = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
3011 4515 5
3012 4516 5      FUNC_VAL_2 = TRMSM_TM_ESCAPE + TRMSM_TM_NOFILTR
3013 4517 5      + TRMSM_TM_TRMNOECHO + TRMSM_TM_NOECHO ;
3014 4518 5
3015 4519 5      RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
3016 4520 5      COB$$RMS_GET ( .RAB, .FUNC_VAL_2, 1, .REST_PTR ) ;
3017 4521 5
3018 4522 5      !+
3019 4523 5      Set TERM_SIZE and TERM_IN_NEXT before possible
3020 4524 5      call to COB$$PARTIAL_SEQ.
3021 4525 5      If user attempts to input data other than a
3022 4526 5      terminator - error.
3023 4527 5      !-
3024 4528 5
3025 4529 5      IF .RAB [RAB$W_RSZ] NEQ 0
3026 4530 5      THEN
3027 4531 6      BEGIN
3028 4532 6      COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;    ! Error.
3029 4533 6      TERM_SIZE = 0 ;
3030 4534 6      END
3031 4535 5      ELSE
3032 4536 5      IF .RAB [RAB$L_STS] EQL RMSS_EOF
3033 4537 5      THEN
3034 4538 5      ! Terminator seen.
3035 4539 5      !+
3036 4540 5      NOTE: When Control Z is typed in as the only
3037 4541 5      input to a $GET it is not recorded in
3038 4542 5      RAB [COB$$B_STV2_LEN] therefore, pull out of
3039 4543 5      loop and let $VERIFY_TERMINATOR handle the ^Z,
3040 4544 5      but first you have to load the ^Z in
3041 4545 5      RAB [COB$$B_STV2_LEN] as this is where
3041 4545 5      $VERIFY_TERMINATOR looks for it.
```



```
3042 4546 5
3043 4547 6
3044 4548 6
3045 4549 6
3046 4550 6
3047 4551 6
3048 4552 5
3049 4553 6
3050 4554 6
3051 4555 6
3052 4556 6
3053 4557 5
3054 4558 4
3055 4559 4
3056 4560 4
3057 4561 4
3058 4562 4
3059 4563 4
3060 4564 4
3061 4565 4
3062 4566 4
3063 4567 4
3064 4568 3
3065 4569 3
3066 4570 3
3067 4571 3
3068 4572 3
3069 4573 3
3070 4574 3
3071 4575 3
3072 4576 3
3073 4577 3
3074 4578 3
3075 4579 2
3076 4580 2
3077 4581 2
3078 4582 1

!-
BEGIN
TERM_SIZE = 1 ;
RAB [COB$$B_STV0_TERM] = CZ ;
LOOK_FOR_TERM = 0 ; ! Set to get out of loop
END
ELSE
BEGIN
LOOK_FOR_TERM = 0 ; ! Set to get out of loop
TERM_SIZE = .RAB [COB$$B_STV2_LEN] ;
TERM_LOC = .RAB [COB$$B_STV0_TERM] ;
END ;
END ; ! End 1 char $GET

!+
!- Check for partial sequence error

IF .RAB [RAB$L_STS] EQL RMSS_PES
THEN
COB$$PARTIAL_SEQ ( .PARAMETERS, .UNIT ) ;
END ; ! End TERM_FROM_DEL=0

!+
!- Now have a Terminator in PUT_HERE. Reset flags. Call
macro to verify Terminator.

TERM_PTR = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
TERM_FROM_DEL = 0 ;
TERM_IN_NEXT = 0 ;
$VERIFY_TERMINATOR ;
END ; ! End Term Loop
END ; ! End COB$$ILLEGAL_TERM
```

## OFFC 00000 COB\$\$ILLEGAL\_TERM:

5E	04	C2	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	4376
	5A	7C	00005	SUBL2	#4, SP	
52	04	AC	D0	CLRQ	LOOK FOR TERM	4417
56	1C	A2	9E	MOVL	PARAMETERS, R2	4425
55	24	A2	9E	MOVAB	28(R2), R6	
57	38	A2	9E	MOVAB	36(R2), R5	
		67	D5	MOVAB	56(R2), R7	
		01	13	TSTL	(R7)	4445
			04	BEQL	2\$	
		5B	D5	RET		
		11	12	TSTL	NO_BELL	4447
		A2	D5	BNEQ	3\$	
	34	0C	12	TSTL	52(R2)	
			0C	BNEQ	3\$	



			0C	AC	DD	00025	PUSHL	FLAGS	4456
				02	DD	00028	PUSHL	#2	
	FD4A	CF		02	FB	0002A	CALLS	#2, COB\$\$RMS_PUT_BYTE	
				02	11	0002F	BRB	4\$	4447
				5B	D4	00031	CLRL	NO BELL	4459
			34	A2	D5	00033	TSTL	52(R2)	4464
				03	13	00036	BEQL	5\$	
				00CE	31	00038	BRW	15\$	
66	18	A2		00	ED	0003B	CMPZV	#0, #16, 24(R2), (R6)	4468
				3D	15	00041	BLEQ	7\$	
			59	18	A2	3C	MOVZWL	24(R2), REST_LEN	4475
			59		66	C2	SUBL2	(R6), REST_LEN	
	58	04	A2	66	C1	0004A	ADDL3	(R6), 4(R2), REST_PTR	4476
				59	D5	0004F	TSTL	REST_LEN	4482
				03	12	00051	BNEQ	6\$	
			59		01	D0	MOVL	#1, REST_LEN	
			50	08	AC	9A	MOVZBL	UNIT, R0	4484
			53	00000000G00	40	D0	MOVL	COB\$\$AL_WRITE_RAB[R0], RAB	
					58	DD	PUSHL	REST_PTR	4485
					59	DD	PUSHL	REST_LEN	
			20	A2	DD	00066	PUSHL	32(R2)	
				53	DD	00069	PUSHL	RAB	
	FC68	CF		04	FB	0006B	CALLS	#4, COB\$\$RMS_GET	
				50	A3	3C	MOVZWL	34(RAB), R0	4491
			22	66	50	C0	ADDL2	R0, (R6)	
				65	0E	A3	MOVZBL	14(RAB), (R5)	4492
			28	A2	0C	A3	MOVZBL	12(RAB), 40(R2)	4493
66	18	A2		10	00	ED	CMPZV	#0, #16, 24(R2), (R6)	4508
					03	12	BNEQ	8\$	
				5A	01	D0	MOVL	#1, LOOK_FOR_TERM	4509
				54	08	AC	MOVZBL	UNIT, R4	4519
					65	D5	TSTL	(R5)	4510
				0A	12	00091	BNEQ	10\$	
	0001827A	8F	08	A3	D1	00093	CMPL	8(RAB), #98938	
				05	12	0009B	BNEQ	11\$	
				01	5A	D1	CMPL	LOOK_FOR_TERM, #1	4511
					53	12	BNEQ	14\$	
	58	04	A2	66	C1	000A2	ADDL3	(R6), 4(R2), REST_PTR	4514
			6E	5240	8F	3C	MOVZWL	#21056, FUNC_VAL_2	4517
			53	00000000G00	44	D0	MOVL	COB\$\$AL_WRITE_RAB[R4], RAB	4519
					58	DD	PUSHL	REST_PTR	4520
					01	DD	PUSHL	#1	
			08	AE	DD	000B8	PUSHL	FUNC_VAL_2	
				53	DD	000BB	PUSHL	RAB	
	FC16	CF		04	FB	000BD	CALLS	#4, COB\$\$RMS_GET	
				22	A3	B5	TSTW	34(RAB)	4529
				0E	13	000C5	BEQL	12\$	
			0C	AC	DD	000C7	PUSHL	FLAGS	4532
				02	DD	000CA	PUSHL	#2	
	FCA8	CF		02	FB	000CC	CALLS	#2, COB\$\$RMS_PUT_BYTE	
				65	D4	000D1	CLRL	(R5)	4533
				BA	11	000D3	BRB	9\$	4529
				5A	D4	000D5	CLRL	LOOK FOR TERM	4550
	0001827A	8F	08	A3	D1	000D7	CMPL	8(RAB), #98938	4536
				09	12	000DF	BNEQ	13\$	
				01	D0	000E1	MOVL	#1, (R5)	4548
			0C	A3	1A	90	MOVB	#26, 12(RAB)	4549



PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------



COB\$ACCEPT  
1-018

COB\$ACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$ILLEGAL\_TERM - Illegal Terminator

E 8  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 84  
(13)

; Routine Size: 415 bytes, Routine Base: \_COB\$CODE + 1037



```
3080 4583 1 %SBTTL 'COB$$CLEAN_UP - Clean up for VAX COBOL'
3081 4584 1 ROUTINE COB$$CLEAN_UP ( PARAMETERS : REF VECTOR,
3082 4585 1     FLAGS
3083 4586 1     ) : NOVALUE =
3084 4587 1
3085 4588 1 ++
3086 4589 1 FUNCTIONAL DESCRIPTION:
3087 4590 1     Perform clean up before returning control to VAX COBOL.
3088 4591 1     Position cursor after FIELD of POTECTED ACCEPT, $PUT to turn
3089 4592 1     attributes off, and determine if ADVANCING is needed.
3090 4593 1
3091 4594 1 FORMAL PARAMETERS:
3092 4595 1
3093 4596 1     PARAMETERS.mlu.ra  Contains data for this routine.
3094 4597 1
3095 4598 1     FLAGS.rlu.v        Screen enhancement flag.
3096 4599 1
3097 4600 1 IMPLICIT INPUTS:
3098 4601 1
3099 4602 1     NONE
3100 4603 1
3101 4604 1 IMPLICIT OUTPUTS:
3102 4605 1
3103 4606 1     NONE
3104 4607 1
3105 4608 1 ROUTINE VALUE:
3106 4609 1
3107 4610 1
3108 4611 1 SIDE EFFECTS:
3109 4612 1
3110 4613 1     NONE
3111 4614 1 --
3112 4615 2 BEGIN
3113 4616 2     LOCAL
3114 4617 2     RAB : REF $RAB_DECL ;
3115 4618 2
3116 4619 2 ++
3117 4620 2     Bind PARAMETERS to other names.
3118 4621 2 --
3119 4622 2
3120 4623 2 $BIND_PARAMETERS ;
3121 4624 2
3122 4625 2 ++
3123 4626 2     Position cursor after FIELD of POTECTED Read.
3124 4627 2     This code is necessary if the # of characters input is less than
3125 4628 2     the # of characters expected. Move cursor the difference of the
3126 4629 2     two numbers.
3127 4630 2     If DEFAULT has been used move cursor the whole length of the expected
3128 4631 2     size.
3129 4632 2 --
3130 4633 2
3131 4634 2 IF .YES_PROTECT
3132 4635 2 THEN
3133 4636 2     BEGIN
3134 4637 2         LOCAL
3135 4638 2         MOVE_CURSOR : INITIAL (0),
3136 4639 2         MOVE_NUM ;
```

```
! Flag
! # of positions to
```



```
3137 4640 3
3138 4641 3      IF .YES_DEFAULT NEQ 0
3139 4642 3      THEN
3140 4643 4          BEGIN
3141 4644 4              MOVE_NUM = .ACC_SIZE ;
3142 4645 4              MOVE_CURSOR = 1 ;
3143 4646 4          END
3144 4647 3      ELSE
3145 4648 3          IF .CHARS_READ LSS .ACC_SIZE
3146 4649 3              THEN
3147 4650 4                  BEGIN
3148 4651 4                      MOVE_NUM = .ACC_SIZE - .CHARS_READ ;
3149 4652 4                      MOVE_CURSOR = 1 ;
3150 4653 4                  END ;
3151 4654 3
3152 4655 3      IF .MOVE_CURSOR NEQ 0
3153 4656 3      THEN
3154 4657 4          BEGIN
3155 4658 4              LOCAL
3156 4659 4                  SPACE_BUF : VECTOR [200,BYTE] ;
3157 4660 4
3158 4661 4              CH$FILL ( BLANK, .MOVE_NUM, SPACE_BUF [0] ) ; ! # of spaces to move
3159 4662 4              COB$$RMS_PUT_BUFFER ( SPACE_BUF [0], .MOVE_NUM, .FLAGS ) ; ! cursor
3160 4663 4          END ;
3161 4664 2      END ;
3162 4665 2
3163 4666 2      !+
3164 4667 2      $PUT to turn attributes off.
3165 4668 2      If no attributes were turned on, there is no need to turn them off.
3166 4669 2      OFF_BUF holds escape sequence to turn attributes off. OFF_LEN holds
3167 4670 2      the length of that sequence.
3168 4671 2      !-
3169 4672 2
3170 4673 2      IF .PUT_FLAG NEQ 0
3171 4674 2      THEN
3172 4675 2          COB$$RMS_PUT_BUFFER ( OFF_BUF [0], .OFF_LEN, .FLAGS ) ;
3173 4676 2
3174 4677 2      !+
3175 4678 2      Determine if ADVANCING is requested.
3176 4679 2      If bit 10 = 0 advancing. If bit 10 = 1 no advancing.
3177 4680 2      Set COB$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
3178 4681 2      statement that advancing/no advancing is required following this
3179 4682 2      ACCEPT statement.
3180 4683 2      !-
3181 4684 2
3182 4685 2      IF (.FLAGS AND V_ADV) NEQ 0
3183 4686 2      THEN
3184 4687 2          COB$$AB_PREV[0] = ACC_DNA
3185 4688 2          ! Signal Do Not Advance
3186 4689 2      ELSE
3187 4690 2          !+
3188 4691 2          Echo carriage return to screen if advancing is called for.
3189 4692 2          !-
3190 4693 3          BEGIN
3191 4694 3              COB$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ;
3192 4695 3              COB$$AB_PREV[0] = ACC_ADV ;
3193 4696 3          END ;
3193 4696 1      END ;
! End of COB$$CLEAN_UP
```



## 01FC 00000 COB\$\$CLEAN UP:

				58	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8	4584
				5E	FF38	CE	9E	00009	MOVAB	COB\$\$AB_PREV, R8	
				56	04	AC	D0	0000E	MOVAB	-200(SP), SP	4617
				38	3C	A6	E9	00012	MOVL	PARAMETERS, R6	4634
						50	D4	00016	BLBC	60(R6), 4\$	4636
					40	A6	D5	00018	CLRL	MOVE_CURSOR	4641
						06	13	0001B	TSTL	64(R6)	
				57	18	A6	3C	0001D	BEQL	1\$	4644
						11	11	00021	MOVZWL	24(R6), MOVE_NUM	4645
1C	A6		18	A6		00	ED	00023	BRB	2\$	4648
				10		0B	15	0002A	CMPZV	#0, #16, 24(R6), 28(R6)	
				57	18	A6	3C	0002C	BLEQ	3\$	4651
				57	1C	A6	C2	00030	MOVZWL	24(R6), MOVE_NUM	
				50		01	D0	00034	SUBL2	28(R6), MOVE_NUM	4652
						50	D5	00037	MOVL	#1, MOVE_CURSOR	4655
						13	13	00039	TSTL	MOVE_CURSOR	
				57	20	6E	00	2C	BEQL	4\$	4661
						6E		00040	MOVC5	#0, (SP), #32, MOVE_NUM, SPACE_BUF	4662
						08	AC	DD	PUSHL	FLAGS	
						57	DD	00044	PUSHL	MOVE_NUM	
						08	AE	9F	PUSHAB	SPACE_BUF	
				FC1E	CF	03	FB	00049	CALLS	#3, COB\$\$RMS_PUT_BUFFER	
						44	A6	D5	TSTL	68(R6)	4673
						0E	13	00051	BEQL	5\$	
						08	AC	DD	PUSHL	FLAGS	4675
						54	A6	DD	PUSHL	84(R6)	
						48	A6	9F	PUSHAB	72(R6)	
				04	FC0B	CF	03	FB	CALLS	#3, COB\$\$RMS_PUT_BUFFER	
					08	AC	0A	E1	BBC	#10, FLAGS, 8\$	4685
						68	05	90	MOVB	#5, COB\$\$AB_PREV	4687
							04	00069	RET		
						08	AC	DD	PUSHL	FLAGS	4693
						7E	D4	0006D	CLRL	-(SP)	
				FB66	CF	02	FB	0006F	CALLS	#2, COB\$\$RMS_PUT_BYTE	
					68	04	90	00074	MOVB	#4, COB\$\$AB_PREV	4694
						04	00077	RET			4696

; Routine Size: 120 bytes, Routine Base: \_COB\$CODE + 11D6



```
3195 4697 1 %SBTTL 'COB$$RPG_CLEAN_UP - Clean up for VAX RPG'
3196 4698 1 ROUTINE COB$$RPG_CLEAN_UP ( FLAGS ) : NOVALUE =
3197 4699 1 ++
3198 4700 1 FUNCTIONAL DESCRIPTION:
3199 4701 1
3200 4702 1 Perform clean up before returning control to VAX RPG.
3201 4703 1
3202 4704 1 FORMAL PARAMETERS:
3203 4705 1
3204 4706 1 FLAGS.rlu.v Screen enhancement flag.
3205 4707 1
3206 4708 1 IMPLICIT INPUTS:
3207 4709 1
3208 4710 1 NONE
3209 4711 1
3210 4712 1 IMPLICIT OUTPUTS:
3211 4713 1
3212 4714 1 NONE
3213 4715 1
3214 4716 1 ROUTINE VALUE:
3215 4717 1
3216 4718 1
3217 4719 1 SIDE EFFECTS:
3218 4720 1
3219 4721 1 NONE
3220 4722 1
3221 4723 1 --
3222 4724 2 BEGIN
3223 4725 2
3224 4726 2 +
3225 4727 2 Determine if ADVANCING is requested.
3226 4728 2 If bit 10 = 0 advancing. If bit 10 = 1 no advancing.
3227 4729 2 Set COB$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
3228 4730 2 statement that advancing/no advancing is required following this
3229 4731 2 ACCEPT statement.
3230 4732 2 -
3231 4733 2
3232 4734 2 IF (.FLAGS AND V_ADV) NEQ 0
3233 4735 2 THEN
3234 4736 2 COB$$AB_PREV[0] = ACC_DNA ! Signal Do Not Advance
3235 4737 2 ELSE
3236 4738 2 BEGIN
3237 4739 2 +
3238 4740 2 Echo carriage return to screen if advancing is called for.
3239 4741 2 -
3240 4742 2 COB$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ;
3241 4743 2 COB$$AB_PREV[0] = ACC_ADV ; ! Signal ADVance
3242 4744 2 END;
3243 4745 2
3244 4746 1 END ; ! End of COB$$RPG_CLEAN_UP
```

0004 00000 COB\$\$RPG\_CLEAN\_UP:  
WORD Save R2

: 4698



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$RPG\_CLEAN\_UP - Clean up for VAX RPG

J 8  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 89  
(15)

04	52	00000000G	00	9E	00002	MOVAB	COB\$\$AB_PREV, R2	:	
	6C		2A	E1	00009	BBC	#42, FLAGS, 1\$	:	4734
	62		05	90	0000D	MOVB	#5, COB\$\$AB_PREV	:	4736
				04	00010	RET		:	
		04	AC	DD	00011	PUSHL	FLAGS	:	4742
			7E	D4	00014	CLRL	-(SP)	:	
FB47	CF		02	FB	00016	CALLS	#2, COB\$\$RMS_PUT_BYTE	:	
	62		04	90	0001B	MOVB	#4, COB\$\$AB_PREV	:	4743
			04	0001E	RET			:	4746

; Routine Size: 31 bytes, Routine Base: \_COB\$CODE + 124E



```
3246 4747 1 $SBTTL 'COB$$FORMAT_FOUR - Format Four'
3247 4748 1 ROUTINE COB$$FORMAT_FOUR ( UNIT      : VECTOR [2,BYTE],
3248 4749 1                                FLAGS,    :
3249 4750 1                                KEY      : REF $STR$DESCRIPTOR
3250 4751 1                                ) =
3251 4752 1
3252 4753 1 ++
3253 4754 1 FUNCTIONAL DESCRIPTION:
3254 4755 1
3255 4756 1     This routine handles VAX COBOL ACCEPT Statement FORMAT FOUR,
3256 4757 1     Control Key.
3257 4758 1
3258 4759 1 FORMAL PARAMETERS:
3259 4760 1
3260 4761 1     UNIT.rbu.va    Array of two unsigned byte integers.
3261 4762 1                    The first byte is the unit number designating the
3262 4763 1                    device from which the string is to be read.
3263 4764 1                    The second byte indicates whether the routine should
3264 4765 1                    abort or return to the calling program.
3265 4766 1                    Byte 2 = 0 - routine will abort on control z
3266 4767 1                               and reprompt on conversion errors.
3267 4768 1                               = 1 - ( AT END )
3268 4769 1                               routine will return to calling program
3269 4770 1                               on control z and reprompt on conversion
3270 4771 1                               errors.
3271 4772 1                               = 2 - ( ON EXCEPTION )
3272 4773 1                               routine will return to calling program
3273 4774 1                               on control z and conversion errors.
3274 4775 1
3275 4776 1     FLAGS.rlu.v    Screen enhancement flag;
3276 4777 1
3277 4778 1     KEY.wt.ds       Destination of the receiving field of the control key.
3278 4779 1
3279 4780 1 IMPLICIT INPUTS:
3280 4781 1
3281 4782 1     NONE
3282 4783 1
3283 4784 1 IMPLICIT OUTPUTS:
3284 4785 1
3285 4786 1     NONE
3286 4787 1
3287 4788 1 ROUTINE VALUE:
3288 4789 1
3289 4790 1
3290 4791 1 SIDE EFFECTS:
3291 4792 1
3292 4793 1     NONE
3293 4794 1
3294 4795 1 --
3295 4796 1
3296 4797 2 BEGIN
3297 4798 2
3298 4799 2 LOCAL
3299 4800 2     RAB      : REF $RAB_DECL,
3300 4801 2     FUNC_VAL,
3301 4802 2
3302 4803 2     TERM_PTR,
```

```
! Read QIO Function Modifiers
! used in item list by RMS
! Pointer to terminator in buffer
```



```
3303 4804 2      NEXT CHAR      : VECTOR [10,BYTE],      ! Buffer to hold terminator sequence
3304 4805 2      LEGAL         : INITIAL (0),           ! = 0 if illegal terminator hit
3305 4806 2      TMASK         : VECTOR [2] ;           ! Longform terminator mask
3306 4807 2
3307 4808 2
3308 4809 2      + Terminator mask - EVERY key is treated as a terminator. Each key pressed
3309 4810 2      is checked for validity as a terminator.
3310 4811 2      Valid terminators are Carriage Return, Tab, Control Z, Arrow keys,
3311 4812 2      PF keys, and the PROFESSIONAL Editing and Top Row Function keys.
3312 4813 2      -
3313 4814 2
3314 4815 2      TMASK [0] = 32 ;
3315 4816 2      TMASK [1] = UPLIT (-1, -1, -1, -1, -1, -1, -1, -1) ;
3316 4817 2
3317 4818 2      +
3318 4819 2      Ring the terminal bell if user requests.
3319 4820 2      -
3320 4821 2
3321 4822 2      IF ( .FLAGS AND V_BELL ) NEQ 0
3322 4823 2      THEN
3323 4824 2          COB$$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
3324 4825 2
3325 4826 2      +
3326 4827 2      Determine FUNC_VAL - QIO Function Modifiers used by RMS $GET Service.
3327 4828 2      Set TRMSM_TM_NOECHO to suppress echoing of input characters to the terminal.
3328 4829 2      Set TRMSM_TM_ESCAPE to allow Escape sequences to act as terminators (Arrow
3329 4830 2      keys and PF keys and the Professional editing and top row function keys).
3330 4831 2      Set TRMSM_TM_NOFILTR to allow this routine to handle the DELETE KEY. (not a
3331 4832 2      valid terminator).
3332 4833 2      Set TRMSM_TM_TRMNOECHO to suppress echoing of the termination character
3333 4834 2      (COB$$$AB_PREV handles advancing / no advancing).
3334 4835 2      -
3335 4836 2
3336 4837 2      FUNC_VAL = TRMSM_TM_ESCAPE + TRMSM_TM_NOFILTR + TRMSM_TM_TRMNOECHO
3337 4838 2                  + TRMSM_TM_NOECHO ;
3338 4839 2
3339 4840 2      $ITMLST_INIT (ITMLST = XAB ITMLST,                ! Item list for $GET
3340 4841 2                  (ITMCO = TRMS_MODIFIERS,
3341 4842 2                  BUFSIZ = 0,
3342 4843 2                  BUFADR = .FUNC_VAL),
3343 4844 2                  (ITMCO = TRMS_TERM,
3344 4845 2                  BUFSIZ = 32,                ! 32 bytes in TMASK
3345 4846 2                  BUFADR = .TMASK[1]) ) ;
3346 4847 2      +
3347 4848 2      RMS $GET - expect only terminators. NOTE: This $GET call is not the
3348 4849 2      same as the call in routine COB$$$RMS_GET.
3349 4850 2      -
3350 4851 2
3351 4852 2      WHILE .LEGAL EQL 0 DO
3352 4853 2          BEGIN                                ! Begin Loop
3353 4854 2
3354 4855 2          RAB = .COB$$$AL_WRITE_RAB [.UNIT[0]] ;
3355 4856 2          RAB [RAB$W_USZ] = 10 ;
3356 4857 2          RAB [RAB$L_UBF] = NEXT_CHAR ;
3357 4858 2          RAB [RAB$V_ETO] = 1 ;
3358 4859 2          RAB [RAB$L_XAB] = XABTRM ;
3359 4860 2          WHILE $GET (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
```



```
3360 4861 3
3361 4862 3
3362 4863 3
3363 4864 3
3364 4865 3
3365 4866 3
3366 4867 3
3367 4868 4
3368 4869 4
3369 4870 4
3370 4871 4
3371 4872 4
3372 4873 3
3373 4874 3
3374 4875 3
3375 4876 3
3376 4877 3
3377 4878 3
3378 4879 3
3379 4880 3
3380 4881 3
3381 4882 3
3382 4883 3
3383 4884 3
3384 4885 3
3385 4886 3
3386 4887 3
3387 4888 3
3388 4889 3
3389 4890 4
3390 4891 4
3391 4892 4
3392 4893 4
3393 4894 4
3394 4895 4
3395 4896 4
3396 4897 5
3397 4898 5
3398 4899 5
3399 4900 4
3400 4901 4
3401 4902 4
3402 4903 4
3403 4904 5
3404 4905 5
3405 4906 5
3406 4907 4
3407 4908 4
3408 4909 4
3409 4910 4
3410 4911 3
3411 4912 3
3412 4913 3
3413 4914 3
3414 4915 3
3415 4916 3
3416 4917 3

IF NOT .RAB [RAB$L_STS]
THEN
+
-
These are special case status that will be handled later.
(See note below for explanation of missing RMSS_TNS)
IF (.RAB [RAB$L_STS] NEQ RMSS_BES AND
.RAB [RAB$L_STS] NEQ RMSS_EOF AND
.RAB [RAB$L_STS] NEQ RMSS_PES AND
.RAB [RAB$L_STS] NEQ RMSS_RTB )
THEN
LIB$STOP (COB$ERRDURACC, 1, .RAB + RAB$C_BLN, .RAB [RAB$L_STS],
.RAB [RAB$L_STV] ) ;

+
-
NOTE: No need for call to COB$$$PARTIAL_SEQ as buffer of 10 bytes
is more than sufficient to hold complete escape sequences.
Most key escape sequences are between 1-4 bytes long.
Status RMSS_TNS, terminator not seen, would signal a need to
call routine COB$$$PARTIAL_SEQ.

TERM_PTR = NEXT_CHAR[0] ;

+
-
Check for legal terminator, then copy it to KEY.

IF .RAB [COB$$$B_STV2_LEN] EQL 1 ! Terminator is one byte
THEN
BEGIN
TERM_PTR = RAB [COB$$$B_STV0_TERM] ;
SELECTONE .RAB [COB$$$B_STV0_TERM] OF
SET
[ CR, ! Carriage Return
TAB ] : ! Tab

BEGIN
CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;
LEGAL = 1 ;
END ;

[OTHERWISE] : ! Error - key not a
! terminator

BEGIN
COB$$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
LEGAL = 0 ;
END ;

TES ;
END

ELSE
IF .RAB [RAB$L_STS] EQL RMSS_EOF
THEN
+
-
CONTROL Z - the status RMSS_EOF is returned
from the $Get Service. ^Z is not stored in
RAB[RAB$STV0_TERM].
```



COBSACCEPT  
1-018

COBSACCEPT - VAX COBOL ACCEPT Statement  
COB\$\$\$FORMAT\_FOUR - Format Four

N 8  
15-Sep-1984 23:54:22  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCEPT.B32;2

Page 93  
(16)

```
3417 4918 3      !-
3418 4919 3
3419 4920 4      BEGIN
3420 4921 4      IF .UNIT [1] EQL 0
3421 4922 4      THEN
3422 4923 4          LIB$STOP ( COB$_EOFON_ACC )      ! Abort
3423 4924 4      ELSE
3424 4925 4          COB$$$CONTROL_Z ( .UNIT, .KEY ) ; ! Return to calling
3425 4926 4          RETURN 0 ;                          ! program.
3426 4927 4      END
3427 4928 3      ELSE
3428 4929 3          !+
3429 4930 3          ! Escape Sequence as Terminator. COB$$$CONTROL_KEY converts
3430 4931 3          ! terminator sequences to COBOL defined sequences and fills
3431 4932 3          ! in KEY parameter if terminator is legal.
3432 4933 3          !-
3433 4934 3      BEGIN
3434 4935 4      IF NOT ( COB$$$CONTROL_KEY (TERM_PTR, .RAB [COB$$B_STV2_LEN],
3435 4936 5          .KEY) )
3436 4937 5      THEN
3437 4938 4          BEGIN
3438 4939 5              ! Error, illegal escape
3439 4940 5              COB$$RMS PUT_BYTE ( RING_BELL, .FLAGS ) ; ! sequence.
3440 4941 5              LEGAL = 0 ;
3441 4942 5          END
3442 4943 4      ELSE
3443 4944 4          LEGAL = 1 ;
3444 4945 4      END ;
3445 4946 2      END ;                                ! End Loop
3446 4947 2
3447 4948 2      !+
3448 4949 2      ! VAX COBOL Version 1 / Version 3 interaction.
3449 4950 2      ! Determine if ADVANCING is requested.
3450 4951 2      ! If bit 10 = 0 advancing. If bit 10 = 1 no advancing.
3451 4952 2      ! Set COB$$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
3452 4953 2      ! statement that advancing/no advancing is required following this
3453 4954 2      ! ACCEPT statement.
3454 4955 2      !-
3455 4956 2
3456 4957 2      IF (.FLAGS AND V_ADV) NEQ 0
3457 4958 2      THEN
3458 4959 2          COB$$$AB_PREV[0] = ACC_DNA      ! Signal- Do Not Advance
3459 4960 2      ELSE
3460 4961 2          !+
3461 4962 2          ! Echo carriage return to screen if advancing is called for.
3462 4963 2          !-
3463 4964 3          BEGIN
3464 4965 3              COB$$RMS PUT_BYTE ( CARR_RET, .FLAGS ) ;
3465 4966 3              COB$$$AB_PREV[0] = ACC_ADV ;      ! Signal- ADVance
3466 4967 3          END ;
3467 4968 2
3468 4969 2      RETURN 1 ;
3469 4970 1      END ;                                ! End of routine COB$$$FORMAT_FOUR
```

01260

.BLKB 3



FFFFFFFF FFFFFFFFF FFFFFFFFF FFFFFFFFF FFFFFFFFF FFFFFFFFF 01270 P.AAS: .LONG -1, -1, -1, -1, -1, -1, -1, -1  
FFFFFFFF FFFFFFFFF FFFFFFFFF FFFFFFFFF FFFFFFFFF FFFFFFFFF 01288

```
01FC 00000 COB$$$FORMAT FOUR:
      58 00000000G 00 9E 00002      .WORD      Save R2,R3,R4,R5,R6,R7,R8      : 4748
      57 00000000G 00 9E 00009      MOVAB      COB$$$AB PREV, R8
      56          FBOC CF 9E 00010      MOVAB      LIB$STOP, R7
      5E          18 C2 00015      MOVAB      COB$$$RMS_PUT_BYTE, R6
      55          55 D4 00018      SUBL2      #24, SP
      04 AE          20 D0 0001A      CLRL      LEGAL
      08 AE          BF AF 9E 0001E      MOVAB      #32, Tmask
      54          08 AC D0 00023      MOVAB      P.AAS, Tmask+4
      54          04 E1 00027      MOVAB      FLAGS, R4
      54          54 DD 0002B      BBC        #4, R4, 1$
      54          02 DD 0002D      PUSHL      R4
      66          02 FB 0002F      PUSHL      #2
      51          5240 8F 3C 00032 1$:      CALLS      #2, COB$$$RMS_PUT_BYTE
      50 00000000' EF 9E 00037      MOVZWL     #21056, FUNC_VAL
      80          80 D4 0003E      MOVAB      XAB ITMLST, $$ITMBLKPTR
      80          51 D0 00040      CLRL      ($$ITMBLKPTR)+
      80          80 D4 00043      MOVAB      FUNC_VAL, ($$ITMBLKPTR)+
      80 00030020 8F D0 00045      CLRL      ($$ITMBLKPTR)+
      80          08 AE D0 0004C      MOVAB      #196640, ($$ITMBLKPTR)+
      80          80 7C 00050      MOVAB      Tmask+4, ($$ITMBLKPTR)+
      53          04 AC 9A 00052      CLRL      ($$ITMBLKPTR)+
      53          55 D5 00056 2$:      MOVZBL     UNIT, R3
      53          03 13 00058      TSTL      LEGAL
      52          00EB 31 0005A      BEQL      3$
      20 52 00000000G 0043 D0 0005D 3$:      BRW        14$
      24 A2          0A B0 00065      MOVAB      COB$$$AL_WRITE_RAB[R3], RAB
      07 A2          0C AE 9E 00069      MOVW       #10, 32(RAB)
      40 A2 00000000' EF 9E 00072      MOVAB      NEXT_CHAR, 36(RAB)
      40          52 DD 0007A 4$:      BISB2      #16, -7(RAB)
      00000000G 00 01 FB 0007C      MOVAB      XABTRM, 64(RAB)
      000182DA 8F 50 D1 00083      PUSHL      RAB
      00000000G 00 52 DD 0008C      CALLS      #1, SYSSGET
      00000000G 00 01 FB 0008E      CMPL      R0, #99034
      50          08 E3 11 00095      BNEQ       5$
      37          50 E8 0009B      PUSHL      RAB
      000181C0 8F 50 D1 0009E      CALLS      #1, SYSSWAIT
      0001827A 8F 2E 13 000A5      BRB        4$
      000181C8 8F 50 D1 000A7      MOVAB      8(RAB), R0
      000181A8 8F 25 13 000AE      BLBS      R0, 6$
      000181A8 8F 50 D1 000B0      CMPL      R0, #98752
      000181A8 8F 1C 13 000B7      BEQL      R0, #98752
      000181A8 8F 50 D1 000B9      CMPL      R0, #98938
      000181A8 8F 13 13 000C0      BEQL      R0, #98938
      000181A8 8F 50 DD 000C2      BEQL      R0, #98760
      000181A8 8F 44 A2 9F 000C7      CMPL      R0, #98760
      000181A8 8F 01 DD 000CA      BEQL      R0, #98728
      000181A8 8F 13 13 000C0      BEQL      6$
      000181A8 8F 50 DD 000C5      PUSHL      12(RAB)
      000181A8 8F 44 A2 9F 000C7      PUSHL      R0
      000181A8 8F 01 DD 000CA      PUSHAB     68(RAB)
      000181A8 8F 01 DD 000CA      PUSHL      #1
```



		00000000G	8F	DD	000CC	PUSHL	#COB\$ ERRDURACC	
67			05	FB	000D2	CALLS	#5, LIB\$STOP	
6E		OC	AE	9E	000D5	MOVAB	NEXT CHAR, TERM_PTR	4882
01		OE	A2	91	000D9	CMPB	14(RAB), #1	4888
			1D	12	000DD	BNEQ	8\$	
6E		OC	A2	9E	000DF	MOVAB	12(RAB), TERM_PTR	4891
50		OC	A2	9A	000E3	MOVZBL	12(RAB), R0	4892
09			50	91	000E7	CMPB	R0, #9	4894
			05	13	000EA	BEQL	7\$	
0D			50	91	000EC	CMPB	R0, #13	
			46	12	000EF	BNEQ	11\$	
50		OC	AC	D0	000F1	MOVL	KEY, R0	4898
04		00	BE	90	000F5	MOVB	@TERM_PTR, @4(R0)	
			46	11	000FA	BRB	12\$	4899
0001827A	8F	08	A2	D1	000FC	CMPL	8(RAB), #98938	4912
			1D	12	00104	BNEQ	10\$	
		05	AC	95	00106	TSTB	UNIT+1	4921
			0B	12	00109	BNEQ	9\$	
		00000000G	8F	DD	0010B	PUSHL	#COB\$ EOFON_ACC	4923
67			01	FB	00111	CALLS	#1, LIB\$STOP	
			49	11	00114	BRB	17\$	
		OC	AC	DD	00116	PUSHL	KEY	4925
0106	C6	04	AC	DD	00119	PUSHL	UNIT	
			02	FB	0011C	CALLS	#2, COB\$\$\$CONTROL_Z	
			3C	11	00121	BRB	17\$	4926
		OC	AC	DD	00123	PUSHL	KEY	4937
7E		OE	A2	9A	00126	MOVZBL	14(RAB), -(SP)	4936
		08	AE	9F	0012A	PUSHAB	TERM_PTR	
00000000G	00		03	FB	0012D	CALLS	#3, COB\$\$\$CONTROL_KEY	
0B			50	E8	00134	BLBS	R0, 12\$	
			54	DD	00137	PUSHL	R4	4940
			02	DD	00139	PUSHL	#2	
66			02	FB	0013B	CALLS	#2, COB\$\$\$RMS_PUT_BYTE	
			55	D4	0013E	CLRL	LEGAL	4941
			03	11	00140	BRB	13\$	4936
55			01	D0	00142	MOVL	#1, LEGAL	4944
		FF	0E	31	00145	BRW	2\$	4852
05	54		0A	E1	00148	BBC	#10, R4, 15\$	4957
68			05	90	0014C	MOVB	#5, COB\$\$\$AB_PREV	4959
			0A	11	0014F	BRB	16\$	
			54	DD	00151	PUSHL	R4	4965
			7E	D4	00153	CLRL	-(SP)	
66			02	FB	00155	CALLS	#2, COB\$\$\$RMS_PUT_BYTE	
68			04	90	00158	MOVB	#4, COB\$\$\$AB_PREV	4966
50			01	D0	0015B	MOVL	#1, R0	4969
			04	0015E	RET			
			50	D4	0015F	CLRL	R0	4970
			04	00161	RET			

; Routine Size: 354 bytes, Routine Base: \_COB\$CODE + 1290

: 3470 4971 1  
: 3471 4972 1 END  
: 3472 4973 0 ELUDOM

! End of module COBSACCEPT



## PSECT SUMMARY

Name	Bytes	Attributes
COB\$DATA	88 NOVEC, WRT, RD ,NOEXE,NOSHR,	LCL, REL, CON, PIC,ALIGN(2)
COB\$CODE	5106 NOVEC,NOWRT, RD , EXE, SHR,	LCL, REL, CON, PIC,ALIGN(2)

## Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	153	1	581	00:00.7
\$255\$DUA28:[COBRTL.OBJ]SMGLIB.L32;1	469	10	2	38	00:00.2

## COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS:COBACCEPT/OBJ=OBJ\$:COBACCEPT MSRC\$:COBACCEPT/UPDATE=(ENH\$:COBACCEPT)

Size: 4745 code + 449 data bytes  
Run Time: 01:09.6  
Elapsed Time: 05:34.6  
Lines/CPU Min: 4290  
Lexemes/CPU-Min: 29151  
Memory Used: 626 pages  
Compilation Complete



0061 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY